Prevention of Pollution Attacks and DoS Attack in Network Coding Using Node Identity Verification Scheme

R. Karthick[1], K. Chandraprabha[2] and B.G.Geetha[3]
[1]PG Scholar, K.S.Rangasamy College of Technology, Tiruchengode, India.
Email: Karthickrak@gmail.com, Mobile No: +91 9600459667.
[2]Assistant professor, K S Rangasamy College of Technology, Tiruchengode, India.
Email: knaprabha@gmail.com
[3]Head of the Department, K S Rangasamy College of Technology, Tiruchengode, India.
Email: hodcse@ksrct.ac.in

Abstract

In recent years, the information maintenance is very difficult. Some attacks may occur on the local system or network based systems. Without security measures and controls in place, our data might be subjected to an attack. Now a day's several attacks are evolved. The most common method of attack involves sending enormous amount of request to server or website and server will be unable to handle the requests and website will be offline for few days or it may take years depends upon the attack. Intra-session network coding is inherently vulnerable to pollution attacks. In this paper, first, we introduce a novel homomorphic MAC scheme called SpaceMac, which allows an intermediate node to verify whether received packets belong to a specific subspace, even if the subspace is expanding over time. Then, we use SpaceMac as a building block to design a cooperative scheme that provides complete defense against pollution attacks: (i) it can detect polluted packets early at intermediate nodes, and (ii) it can identify the exact location of all, even colluding, attackers, thus making it possible to eliminate them. Our scheme is cooperative: parents and children of any node cooperate to detect any corrupted packets sent by the node, and nodes in the network cooperate with a central controller to identify the exact location of all attackers. We Implement SpaceMac in both C/C++ and Java as a library, which we make publicly available. Our evaluation on both a PC and an Android device shows that the SpaceMac algorithms can becomputed quickly and efficiently and that our defense scheme has low computation overhead and significantly lower communication overhead than those of state-of-the-art schemes.

***Keywords*: Communication networks, network coding, cooperative systems, communication system security, cryptographic protocols, message authentication, pollution attacks.**

I.INTRODUCTION

The Network coding paradigm advocates that intermediate nodes in a network should mix incoming packets instead of simply forwarding them, and receivers should decode to obtain the original packets. In this work, we consider networks that employ intra-session linear network coding. An inherent weakness of network coding is that it is particularly vulnerable to pollution attacks. Malicious nodes can inject corrupted packets into a network, which get combined and forwarded by downstream nodes, thus causing a large number of corrupted packets propagating in the network. This wastes resources and eventually prevents the decoding of the original packets at the receivers. Thedetrimental effect of pollution attacks has been shown through both theoretical analysis as well as experimentation. Proposed defense mechanisms against pollution attacks can be classified into three categories: error correction, attack detection, and locating attackers. In this paper, we are interested in the latter two approaches. In particular, we set out to design a complete defense system that can not only detect polluted packets in a timely manner but can also accurately locate and eliminate all, even colluding, attackers. This allows for dealing with an attack early and at its root. To the best of our knowledge, none of the existing defense mechanisms can provide this level of protection. To this end, we first propose a novel homomorphic message authentication code (MAC) scheme for expanding spaces, called SpaceMac. SpaceMac allows a node to verify if its received packets belong to a specific subspace, even if the subspace is expanding over time. Then, we design a novel cooperative defense system which includes both a detection scheme and a locating scheme, using SpaceMac as their building block. Our detection scheme relies on SpaceMac to force intermediate nodes to send only linear combinations of packets that they actually receive from their parents. Parents and children of any intermediate node cooperate to detect corrupted packets sent by the intermediate node. Our locating scheme uses

SpaceMac to force nodes in the network to truthfully cooperate with a central controller so that the controller can exactly locate the pollution attackers. Finally, by leveraging multiple generations, our scheme is able to deal with an arbitrary number of colluding attackers. The main contributions of this paper are the following:

The design and implementation of SpaceMac:

We describe the construction of SpaceMac and provide a formal security proof for the construction. We implement SpaceMac in both C/C++ and Java as a ready-to-use library, which we make available online. Our Java implementation is compatible with the current Android OS (Android 2.2 Froyo).

The design of a novel cooperative defense system based on SpaceMac:

To the best of our knowledge, our defense system is the first that meets all of the following requirements:

(i) it can provide timely in-network detection, (ii) it can exactly locate all pollution attackers, (iii) it can deal with an arbitrary number of colluding attackers, and (iv) it has low communication and computation overhead. We have extensively evaluated the computation overhead of SpaceMac's algorithms and the computation and communication overhead of our defense system, through implementation in both C/C++ and Java, and on both a PC and an Android device. Our evaluation results show that for a practical parameter setting, all three algorithms of SpaceMac (Mac, Combine, and Verify) can be computed efficiently (requiring 64 KB of memory in C/C++ or 128 KB in Java) and also quickly on a PC (<28 μs in C/C++) and even on a smartphone (<2.3 ms). Evaluation results also demonstrate that for the same practical parameter setting, when implementing our defense system, nodes in the network introduce very small computational delay per packet (in the order of sub-millisecond on the PC and millisecond on the smart phone). Moreover, our defense system introducesvery low communication overhead per packet (2%), significantly less than those of state-of-the-art schemes.

II.RELATED WORK

Attack Detection

The scheme in HomMac, relies on cover free set systems for pre-distributing keys to provide in-network detection, and thus is only collusion resistant. Furthermore, it is susceptible to tag-pollution attacks, where malicious nodes tamper with some subset of tags of a packet. The scheme is collusion resistant as well as resistant against tag-pollution attacks; however, it requires time synchronization among nodes in the network. Both schemes have low computation overhead since they only require simple addition and multiplication operations at intermediate nodes for both combining MAC tags and tag verification. Our SpaceMac scheme is inspired by and generalizes HomMac, with the difference that SpaceMac allows intermediate nodes to sign subspaces that *expand* over time, while HomMac allows to sign only fixed subspaces. In this paper, (i) we provide a much more efficient construction of SpaceMac and (ii) we show that the ability to authenticate expanding subspaces can also be utilized to provide in-network detection.

Locating Attackers

Early work proposed that the subspace properties of randomized network coding to locate pollution attackers. The main observation was that packets sent by a node have to belong to the space spanned by source packets and also the space spanned by the packets the node receives from its parents. Using this observation, in a general network topology having a single attacker, the authors can locate the attacker with an uncertainty of at most two nodes; when there are multiple attackers. The uncertainty is within a set of nodes including the attackers and their parents and children. Our scheme builds on and significantly improves this work: we make it possible to pinpoint the exact location of the attackers, even in the case where there are multiple colluding attackers, thereby allowing for the removal of all attackers.

Wang *et al.* introduced a light-weight non-repudiation protocol ensuring that (i) a malicious node that injected a polluted packet cannot deny its behavior and (ii) a malicious node cannot disparage any innocent node. They built a defense scheme based on the protocol to identify malicious nodes. Moreover, because the success of the locating scheme relies on the successful reception of the checksums at every node, this scheme is vulnerable to colluding attackers. The scheme is also unable to locate all the attackers. We use the non-repudiation protocol as a building block in our locating scheme. However, our scheme is able to locate all, even colluding attackers, without the need of checksums.

Design Goals

With this threat model in mind, we set out to design adefense system with the following capabilities and properties:

In-network Detection

Any intermediate node in the network should be able to detect the attack as soon as its

malicious parent sends it a corrupted packet. This prevents corrupted packets from polluting the downstream edges.

Exact Locating

The location of all pollution attackers should be precisely identified. This allows for the removalof the attackers from the network.

Arbitrary Collusion Resistance

The defense system should be able to detect attacks and remove the attackers from the network even when they collude.

Low Overhead

The system should require a small amountof computing from intermediate nodes and should not introduce a large amount of traffic, *e.g.*, bandwidth of the MAC tags, to the network. To achieve the above goals, we design a defense system which consists of two main components: the detection scheme and the locating scheme. The detection scheme provides in-network detection while the locating scheme provides exact locating. Both the detection scheme and locating scheme impose low computation as well as low communication overhead. The defense system as a whole is arbitrarily collusion resistant.

III. KEY OBSERVATIONS AND APPROACH OVERVIEW

In-Network Detection

The detection works by first establishing shared secret MAC keys between the source and the intermediate nodes.Then, using these secret keys, the source node can sign the fixed source space and the intermediate nodes can verify if their received packets belong to the source space. Our detection scheme leverages a different observation: A packet sent by an intermediate node must belong to the space spanned by all packets that it received from its parents. A packet sent by *C* must belong to the space spannedby the packets it received from its parents: *A* and *B*; otherwise *C* must be polluting the network.

Exact Locating

Leveraging the cooperation among nodes in the network anda central controller, when there is a single pollution attacker, its locationcan be narrowed down to a set of at most two nodes. This is done by analyzing the polluted edges identified based onthe incoming subspaces reported by all nodes to the central controller. When there are multiple attackers in a general network topology, the number of suspected nodes increases to include the attackers and their parents and children.

Our key observation here is that the uncertainty aboutthe location of the attackers originates from the fact that the attackers can lie about their received spaces. Therefore, by ensuring that all nodes in the network cannot lie about their received spaces, we can exactly locate the attackers.

IV. THE CONSTRUCTION OF SPACEMAC

The key difference between SpaceMac and HomMac is the security property that SpaceMac brings: SpaceMac allows for signing spaces that expand over time, while HomMac only allows for signing fixed spaces. This is directly reflected in the difference between the security games of SpaceMac and HomMac, and consequently in the difference between the constructions of the two schemes.

A (*q, n,m*) homomorphic MAC scheme is defined by three probabilistic, polynomial-time algorithms: Mac, Combine, and Verify. The Mac algorithm generates a tag for a given vector; the Combine algorithm computes a tag for a linear combination of some given vectors and a tag is a valid tag of a given vector.

- Mac(*k*, id, **y**):
  - Input: a secret key *k*, the identifier id of the source space and a vector **y** .
  - Output: tag *t* for **y**.
- Combine ((**y**1, *t*1, *α*1),…., (**y***p*, *tp*, *αp*)):
  - Input: *p* vectors **y**1,….,**y***p*, their tags *t*1,…, *tp* under key *k* and their coefficients.
  - Output: tag *t* for vector **y** = *αi* **y***i*.
- Verify(*k*, id, **y**, *t*):
  - Input: a secret key *k*, the identifier id of the source space, a vector **y** and its tag *t*
  - Output: 0 (reject) or 1 (accept)

V. DETECTION SCHEME

For ease of presentation, we describe the detection scheme within the scope of a single generation, *i.e.*, considering a single source space id.

Assumptions

We assume that there is a controller who knows the complete topology of the graph. The controller could be the source itself. We further assume that each node*N* shares with the controller a pair of secret keys (*k*1*N*, *k*2*N*). These keys can be established with a public key infrastructure.

Bootstrapping

First, for every intermediate node *N*, the controller determines the key, which is secret to *N* itself and is used by the parents and children of *N* when using SpaceMac. Each node can serve as either a parent or a child. Hence, each node, depending on its position in the network must know a different set of keys to participate in the detection scheme. The controller then sends to each node *N* a bootstrapping packet consisting of the set of keys that are necessary for it to participate in the detection scheme.

Sending and Coding

Before sending out each source packet, **v***i*, the source *S* calculates an *end-to-end tag*, *tk* using the Mac algorithm of SpaceMac with key *k and tag tk. S* then attaches this tag to every source packet and sends **w***i*. The packets traversing the network are linear combinations of **w***i*'s instead of **v***i*'s. *P* needs to calculate a *helper tag* which helps the children of *N* to detect corrupted packets sent by *N*. In particular, before sending **y** to *N*, *P* needs to calculate a MAC tag.

Receiving and Verification

When a node *N* receives from its parent *P* a packet *P* and the probability that *P* can forge a valid tag *tk*, when **y** is outside of its received space. As soon as *N* receives a corrupted packet from *P*, with high probability, *N* is able to detect the attack immediately. If *N* is a receiver, it further verifies the end-to-end tag using key *k*. Since none of the malicious intermediate node knows $k*$, if **w** is outside of the source space. This second level of verification provides a detection mechanism in the presence of colluding adversaries.

VI. LOCATING SCHEME

Reporting

For each received subspace *P*, from a parent *P*, node *N* may report a randomly chosen packet **y***r*, of the space instead of the space itself; and by checking if **y***r*, the controller can determine to identify the polluted edges.

Using SpaceMac

We use SpaceMac to prevent nodes from lying about their received spaces as follows. Whenever*P* sends a vector **y***i* to *N*, it generates a tag, *t***y***i*, of **y***i* using the Mac algorithm with a secret key shared by *P* andthe controller. Then, when *N* reports **y***r*, if **y***r* is a linear combination of vectors that it received from *P*, **y***i*'s, then *N* can generate a valid tag of **y***r* by using the Combine algorithmon the tags of **y***i*'s that it received.

Non-Repudiation Transmission Protocol

SpaceMac forces nodes to report only received subspaces are computationally difficult to forge valid tags otherwise. However, it does not prevent a malicious node from sendinginvalid tags to its children to prevent the children from reporting polluted spaces.

Locating the Attackers

To locate the attackers, we proceed by partitioning the edges into two set: the set of polluted edges, *Ep*, and non-polluted edges, *Es*, then analyzing the nodes with respect to the identified *Ep* and *Es*. Preventing nodes from lying, our scheme produces an unambiguous partitioning of *Ep* and *Es*, which helps to precisely locate the attacker. In particular, the adversary isalways the node that has no incoming edge belonging to *Ep* but has at least one outgoing edge belonging to *Ep*. Due to limited space, we refer the reader to a detailed description of the locating scheme and (ii) adetailed analysis of the security of the defense system, wherewe discuss (ii-a) multiple colluding adversaries, (ii-b) tag pollution attacks, (ii-c) denial of service attacks, as well as (ii-d) malicious receiver scenarios.

VII. PERFORMANCE EVALUATION

Detection scheme

For the online overhead per packet, *i.e.*, the additional bandwidth required per packet sent in the network, our detection scheme requires each packet to carrythree SpaceMac tags: an end-to-end tag, a helper tag, and averification tag. Our communication overhead is fixed, regardless of the network topology.

Locating Scheme

For each packet received, each nodeverifies δ tags using the Verify algorithm. For each packet itsends out, each node needs to compute λ tags using the Mac algorithm. The total overhead is therefore $(\delta + \lambda)(n + 2m)$ number of multiplications.

Combined Scheme

The overall computation overhead per packet per node of our combined scheme is $(3 + \delta + \lambda)(n + 2m) + w$. We implement multiplication in F28 by creating anoffline multiplication table, storing all 216 products of pairs of elements in this field. The table only occupies about 64KB in C/C++ and 128 KB in Java. The numbers reported areaveraged over 106 multiplications. Our PC has a quad-core 2.8 Ghz processor and 32 GB of RAM, and our Android device isa Samsung Captivate with a single 1 Ghz processor and 512MB RAM. The computational latency of our detection scheme is in the same order of magnitude as the other two detection schemes. We

note, however, that our detection scheme achieves all of the desired properties: in-network detection, arbitrary collusion resistance and tag-pollution resistance without the need of time synchronization. The latency of our combined detection-locating scheme is about 10 times higher than that of our detection scheme. This is the trade-off when one wants to locate and eliminate all attackers.

VIII. CONCLUSION

In this work, we introduce a novel homomorphic MAC scheme for expanding spaces, called SpaceMac, and we propose a cooperative defense system against pollution attacks built on SpaceMac. To the best of our knowledge, our system is the first that can provide both in-network detection and exact locating of the attackers. In addition, our system is collusion resistant and tag-pollution resistant. Our evaluation results using real implementation in C/C++ and Java on multiple devices demonstrate that our defense system incurs both low communication and low computation overhead. We implemented SpaceMac as a ready-to-use library and make it available online.

REFERENCES

[1] Sidharth Jaggi, Michael Langberg, Tracey Ho (2003) "Correction of Adversarial Errors in Networks", IEEE INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY (ISIT), PAGE 368, YOKOHAMA.

[2] Ralf Koette and Muriel Médard (2009)," An Algebraic Approach to Network Coding", Science Direct on Advances in Control Engineering and Information Science.

[3] Anh Le and Athina Markopoulou (2009)," Locating Byzantine Attackers in Intra-Session Network Coding using SpaceMac", IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 4.

[4] Mahdi Jafarisiavoshani, Christina Fragouli, Suhas Diggavi (2006), "Subspace Properties of Randomized Network Coding", IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 3.

[5] Shuo-Yen Robert Li, Raymond W. Yeung and Ning Cai (2004)," Linear Network Coding" , IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 49, NO. 2.

[6] Dan Boneh, David Freeman, Jonathan Katz and Brent Waters (2005)," Signing a Linear Subspace: Signature Schemes for Network Coding", IEEE Trans. on Information Theory 54, 2596–2603.

[7] Shweta Agrawal1 and Dan Boneh (2007), "Homomorphic MACs: MAC-based Integrity for Network Coding", IEEE/ACM Transactions on Networking, 11:782{795.

[8] Ning Cai and Raymond W. Yeung," Secure Network Coding on a Wiretap Network" IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 46, NO. 4, JULY 2008.

[9] Jing Dong, Reza Curtmola and Cristina Nita-Rotaru (2005) "Practical Defenses Against Pollution Attacks in Intra-Flow Network Coding for Wireless Mesh Networks" Science Direct on the journal of systems and software, 75(2005)63.

[10] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," IEEE TRANSACTIONS ON INFORMATION THEORY, vol. 46, no. 4, pp. 1204–1216, 2000.

[11] D. Silva, F. R. Kschischang, and R. Koetter, "A rank-metric approach to error control in random network coding," in IEEE INFORMATION THEORY WORKSHOP, 2007.

[12] J. S. Plank, "Fast galois field arithmetic library in C/C++," University of Tennessee, Tech. Rep. UT-CS-07-593, March 2007.