Security Analysis of a Protocol for Pollution Attack Detection

Kiattikul Sooksomsatarn, Ian Welch, Winston K. G. Seah

School of Engineering and Computer Science
Victoria University of Wellington
PO Box 600, Wellington 6140, New Zealand
{kiattikul.sooksomsatarn,ian.welch,winston.seah}@ecs.vuw.ac.nz

***Abstract*—Network coding is a technique for maximizing the use of available bandwidth capacity. This is achieved by having nodes not just forwarding packets but combining several incoming packets into a single outgoing packet for transmission. Unfortunately, network coding is vulnerable to pollution attacks where a single malicious node can disrupt the operation of the complete network. Several protocols to detect pollution attacks have been proposed in the literature. In this paper we describe a new pollution attack detection protocol that extends the existing SpaceMac protocol. This paper describes how we have modeled the protocol in order to carry out a security analysis and presents the results of that analysis.**

I. Introduction

With the rise of bandwidth intensive applications such as on-demand video or video conferencing, bandwidth conservation has attracted attention. One way proposed to save bandwidth is a technique for maximizing the use of available bandwidth called network coding [1]. Network coding allows multiple packets to be transmitted using a smaller number of packets thereby increasing throughput. In a network coding protocol, a common single base station transmits data from a common single base station to intermediate stations where it is kept and sent out to the final destination or to any other intermediate stations at a later time. In a traditional multi-cast network, the intermediate stations receive a packet and forward it to the next node. Under network coding, nodes can combine a number of incoming packets of data that they have received into one or several outgoing packets. At the receiver, the original packets are extracted from the combined packets. Successful encoding and extracting requires all nodes to share a coding algorithm that can be used for encoding and decoding of packets.

Even though applying network coding is an attractive idea, it is vulnerable to pollution attacks where a malicious node can inject bad packets into the network that prevent the receiver from successfully extracting the original packets. Alternatively, legitimate packets can be turned into bad packets by a forwarding node performing packet modification. A single malicious node can cause a widespread Denial-of-Service (DoS) because other nodes will *unquestionably* combine the bad packets with good packets and forward the resulting corrupted packet to downstream nodes. The detrimental effect of pollution attacks has been shown through both theoretical analysis [2] as well as experimentation [3], [4].

Section II introduces the new security protocol in the context of previous pollution attack detection schemes, Section III describes how we model this protocol for the purposes of security analysis, Section IV provides an overview of how this model is used for security analysis and Section V concludes the paper.

II. Related Work

The simplest pollution attack detection scheme is end-to-end attack detection where the sender adds message authentication codes (MACs) as tags to each packet and the receiver simply validates each MAC to determine that it has been received correctly. Should the packets be corrupted, the receiver

asks for retransmission of the corrupted packets. This scheme does not add any load to intermediate nodes but may lead to wasted bandwidth because even when the corruption occurs far upstream, the packets must reach the receiver to trigger a retransmission. This has led to a preference for hop-by-hop schemes where MACs are checked at each hop allowing early detection of corruption and reducing wastage but requiring intermediate nodes to do extra work and also the use of cryptography to ensure that nodes cannot fabricate MACs to evade the pollution detection mechanism (a tag-pollution attack).

Digital signatures can be used as MACs that can be tied to a node but this requires each intermediate node to verify the integrity of each packet as it is received, combining multiple received packets together and recomputing a digital signature for the new combined packet. In order to simplify this process, detection schemes moved to use homomorphic cryptography which allows the combination of packets and checking of integrity without the need for decryption [5], [9], [10]. The high cost of homomorphic cryptography was problematic and later schemes proposed the use of algorithms based upon the use of linear subspaces to construct MACs because they it only requires simple addition and multiplication operations at intermediate nodes for both combining MAC tags and to verify the MACs [8].

We have developed a new protocol based upon Cooperative SpaceMac[11], [12] (currently the best of the protocols) to detect pollution attacks. The new protocol adds loose key synchronization and improves the cooperative aspects of the protocol to improve its ability to detect attacks at an early stage. The modeling of this protocol is described in the next Section.

III. Modelling the Proposed Protocol

This section presents a high level definition of our proposed security protocol that could be implemented in various ways as long as they fulfill the requirements of the description. We describe the system model representing all these agents involved in the protocol (a high-level overview of the protocol), the threat model, the security goals of the protocol are identified and we discuss how we model the honest agents as well as the attackers. A feature of this model is the use of the process algebra Communicating Sequential Processes (CSP) [13] [14] to represent the possible actions of parties executing the protocol.

Overall, the intention of modeling the proposed protocol is to validate the protocol in terms of its security properties using formal techniques. In particular, this model would allow a security analysis using the model checker Failures-Divergence Refinement (FDR) [15].

A. Overview of Pollution Attacks Detection Protocol

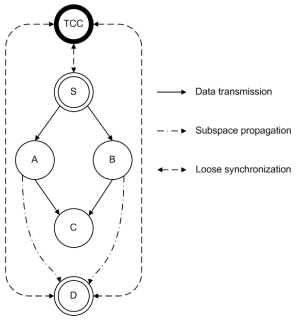

Fig. 1. Agents Diagram

Figure 1. shows all these participants involved in the protocol. The roles of the $Agents$ in the protocol are then defined as follows: (1) *Trusted Central Controller(TCC)* (nicknamed *Trent(T)*) cooperates with in-network nodes to defend against pollution attacks. Trent has two main mechanisms: generating authentication tags and verifying reports created by insiders using loose synchronization. If a recipient has a valid tag signed by TCC then they can be confident in the authentication of origin of a sender. (2) *Source(S)* is a single node that multicasts packets with their encrypted authentication information to a destination via intermediate nodes. Typically, *Source* and *Trent* can be located on the same server so that to be easy for manipulation. (3) *Destination(D)* is an end receiver, or an end user who requests the transmission of the packets. (4) *Intermediate nodes* are all the nodes except for

those playing the roles of 1,2 and 3. *Intermediate nodes* each have *Parent* and *Child* nodes. In a given session, a *Parent* can be multiple senders but *Child* is only a single receiver.

The main features of our new pollution attack detection protocol include: (1) Cooperation of nodes with the *Source* to create initial public/secret keys and initiate the collection of necessary information. (2) Generation of authentication tags at the beginning of the transmission. The authentication tags are generated and signed depending on content of outgoing packets. The authentication tags signed by the *Source* are called *SourceSpace*. (3) Allows a recipient to combine and redistribute received packets and create a propagated *SubSpace*. If a recipient wishes to redistribute the packets they have received from sender(s), they can take the role of a sender and send it to another recipient. (4) Provides a method for a recipient to check the authentication of origin of a packet they received, even from untrusted senders.

B. Threat Model

In an open environment we must anticipate that attackers exist who want to inject faked messages. The injected messages can be malicious and represent a pollution attack. In this paper, we call our attacker is *Paul*. Paul can be an individual (a single attacker), a coalition of a group of attackers (multiple attackers), and a legitimate node in the protocol (an insider). We assume that the adversary follows a Dolev-Yao threat model [16]. We further assume that there are a single source node $\mathcal{S}$ that multicasts packets to a set of end receivers $\mathcal{R}$ which they are trustworthy. Set of edges or links $\mathcal{E}$ between nodes are untrusted. Set of intermediate nodes $\mathcal{I}$, denotes that $\mathcal{I} = \mathcal{V} - \{\mathcal{R} \cup \{\mathcal{S}\}\}$, can be compromised, collude and lie about their authentication information.

C. Security Goals

We have grouped the possible pollution and tag-pollution attacks into the following categories:

1) Colluding: The malicious attacker tries to inject faked packets into the network with cooperation of an insider node. The insiders allow the attacker to use their valid tag to create a new valid tag. Colluding can occur in two scenarios: colluding between *Parents* and colluding between *Parent* and *Child*.

2) Packet Sniffing: The malicious attacker sees a valid tag on the network link and uses it to generate a new forged tag.

3) Spoofing: The attacker claims to be the legitimate insider. The attacker has to trick *Trent* or *Source* to have a desired packet and its valid tag.

4) Fabrication: The attacker tries to produce a valid tag for a packet without knowledge of the secret key for the packet.

D. Modelling the Honest Agents

We now describe how we can model the honest agents running the protocol as CSP processes. To simplify our work we use the *Casper* compiler to automate the generation of the CSP models from a more abstract description written by the user. We give a parameterised process $Initiator(A, k_A)$ to represent an agent a running the protocol as initiator, and using session key k_A. The process starts by receiving a message m from the environment, telling it with whom to run the protocol. It then sends an appropriate message 1 $m1$, and receives back an appropriate message 2 $m2$ containing an arbitrary value for nonce of responder n_B.

The definition of the responder is similar: the process $Responder(B, n_B)$ represents agent B running the protocol as responder using nonce n_B. The responder starts by receiving a message 1 $m1$, from an arbitrary agent a and containing an arbitrary session key k. It then sends back the corresponding message 2 $m2$.

We consider a small system, comprising Alice acting as initiator, using key k_A, and Bob acting as responder, using nonce n_B. The two agents do not communicate directly: we arrange below for all communications to go via the attacker. We model this as an interleaving.

E. Modelling the Attacker

We now describe how we can model the attacker. The main issue is modelling which messages the attacker is able to understand and to create. We need

to keep track, therefore, of which sub-messages of protocol messages the attacker knows; we term these $Facts$. Essentially the attacker can learn facts through observation or simple deductions.

If the Attacker knows a fact f and a key k then he can encrypt f with k; if he knows an encrypted message and the corresponding decryption key, he can perform the decryption to obtain the body; if he knows a collection of facts, he can concatenate them together; if he knows a concatenation, he can split it up into the individual components.

IV. Security Analysis

We do not have space to present the security analysis here. However, the general approach was to consider a small system running the protocol: we include a single initiator Alice, who will use the session key Ka, and a single responder Bob, who will use the secret Sb. We also include a pollution attacker, Paul, who has complete control over the network. We used FDR to explore the potential states that the system could reach and to identify any possible states that might violate the security properties established earlier. We did not find any attacks that would compromise the protocol.

V. Conclusion and Discussion

We have provided an overview of how we can model our new pollution attack detection scheme it for the purpose of security analysis. Analysis using model checking did not identify any problems in our protocol. However, we should take the absence of problems not as proof that there are none. This is because, in practice, the protocol tends to be more complicated that our running protocol. For example: the messages used in the protocol might be more complex, etc. Therefore in further analysis we intend to also apply proof techniques to support the analysis done so far.

References

[1] R. Ahlswede, C. Ning, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[2] M. Kim, L. Lima, F. Zhao, J. a. Barros, M. Médard, R. Koetter, T. Kalker, and K. J. Han, "On counteracting byzantine attacks in network coded peer-to-peer networks," *IEEE J.Sel. A. Commun.*, vol. 28, pp. 692–702, June 2010.

[3] J. Dong, R. Curtmola, and C. Nita-Rotaru, "Practical defenses against pollution attacks in wireless network coding," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 1–31, jun 2011. [Online]. Available: http://doi.acm.org/10.1145/1952982.1952989

[4] C. Gkantsidis and P. Rodriguez, "Cooperative security for network coding file distribution," in *the Proceddings of IEEE INFOCOM*, 2006.

[5] M. Krohn, M. Freedman, and D. Mazieres, "On-the-fly verification of rateless erasure codes for efficient content distribution," in *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, may 2004, pp. 226–240.

[6] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger, "Byzantine modification detection in multicast networks with random network coding," *Information Theory, IEEE Transactions on*, vol. 54, no. 6, pp. 2798 –2803, june 2008.

[7] F. Zhao, T. Kalker, M. Medard, and K. J. Han, "Signatures for content distribution with network coding," in *IEEE International Symposium on Information Theory 2007. ISIT 2007.*, june 2007, pp. 556–560.

[8] E. Kehdi and B. Li, "Null keys: Limiting malicious attacks via null space properties of network coding," in *INFOCOM 2009, IEEE*, april 2009, pp. 1224 –1232.

[9] S. Agrawal and D. Boneh, "Homomorphic macs: Mac-based integrity for network coding," in *Proceedings of the 7th International Conference on Applied Cryptography and Network Security*, ser. ACNS '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 292–305.

[10] P. Zhang, Y. Jiang, C. Lin, H. Yao, A. Wasef, and X. Shen, "Padding for orthogonality: Efficient subspace authentication for network coding," in *INFOCOM, 2011 Proceedings IEEE*, april 2011, pp. 1026 –1034.

[11] A. Le and A. Markopoulou, "Locating byzantine attackers in intra-session network coding using spacemac," in *Network Coding (NetCod), 2010 IEEE International Symposium on*, june 2010, pp. 1 –6.

[12] ——, "Cooperative defense against pollution attacks in network coding using spacemac," *Selected Areas in Communications, IEEE Journal on*, vol. 30, no. 2, pp. 442–449, february 2012.

[13] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe, "A theory of communicating sequential processes," *J. ACM*, vol. 31, no. 3, pp. 560–599, Jun. 1984. [Online]. Available: http://doi.acm.org/10.1145/828.833

[14] A. W. Roscoe, C. A. Hoare, and R. Bird, *The theory and practice of concurrency*. Prentice Hall Englewood Cliffs, 1998, vol. 1.

[15] N. Brownlee, "Formal systems (europe) ltd. failures-divergence refinement. fdr2 user manual. available at http://www.formal.demon.co.uk/fdr2manual/index.html," in *Blount MetraTech Corp. Accounting Attributes and Record Formats http://www.ietf.org/rfc/rfc2924.txt*, 2000.

[16] D. Dolev and A. C. Yao, "On the security of public key protocols," *Information Theory, IEEE Transactions on*, vol. 29, no. 2, pp. 198–208, 1983.