Achieving Multicast Stream Authentication Using MDS Codes

Christophe Tartary and Huaxiong Wang

Centre for Advanced Computing, Algorithms and Cryptography
Department of Computing
Macquarie University
NSW 2109 Australia
{ctartary, hwang}@ics.mq.edu.au

Abstract. We address the multicast stream authentication problem when the communication channel is under the control of an opponent who can drop, reorder or inject data. In such a network model, packet overhead and computing efficiency are important parameters to be taken into account when designing a multicast authentication protocol. Our construction will exhibit three main advantages. First, our packet overhead will only be a few hashes long. Second, we will exhibit a number of signature verifications to be performed by the receivers which will turn to be $O(1)$. Third, every receiver will still be able to recover all the data packets emitted by the sender despite losses and injections occurred during the transmission of information.

Keywords: stream authentication, polynomial reconstruction, erasure codes.

1 Introduction

Broadcast communication is an essential mechanism to disseminate digital media from a single sender to a large audience via a public channel such as the Internet. The applications cover a broad area including digital radio, air traffic control as well as software updates for instance. Unfortunately large-scale multicasts prevent lost content from being redistributed since the loss of any piece of the data stream can cause a flood of retransmission requests at the sender. In addition the network can be under the control of malicious users performing harmful actions on the data stream. Therefore the security of a broadcast protocol relies on two aspects: the network properties and the opponents' computational power. Investigations concerning unconditionally secure protocols have been made in [5, 7, 38]. Unfortunately either these schemes can only be used for a single authentication or they have too large storage requirements for practical applications. In this work, we will consider that the opponents have bounded computational powers.

Applications like digital TV or stock quotes suggest that the data stream can be large and eventually infinite. Nevertheless receivers must be able to authenticate collected information within a short period of delay upon reception. Since many protocols will transfer private or sensitive content, non-repudiation of the sender is required for most

D. Pointcheval, Y. Mu, and K. Chen (Eds.): CANS 2006, LNCS 4301, pp. 108–125, 2006.

of them. Unfortunately signing each packet[1] is impracticable as digital signatures are generally time expensive. In addition bandwidth limitations prevent k-time signatures [36] from being used due to their large size. That is why a general approach is to generate a single signature and to amortize its communication and computation overheads over several packets using hash functions for instance.

By appending the hash of each packet to several followers according to some specific patterns, Perrig et al. [32, 33], Golle and Modadugu [10] and Miner and Staddon [24] designed schemes dealing with packet loss. One signature was generated from time to time and was always assumed to be received. In these contributions, authors modeled the network packet loss by a k-state Markov chain [8, 31, 43] and provided bounds on the packet authentication probability. Gao and Yao [9] proposed to use online/offline signature to speed up signing and verifying time for these schemes. Unfortunately all these protocols rely on reception of signed packets. Since networks like the Internet only provide a best effort delivery, it narrows the range of applications of these schemes.

To overcome this problem one solution is to split the signature into k smaller parts where only ℓ of them ($\ell < k$) are enough for recovery. Along this line, several schemes were developed [1, 27, 28, 29, 30] but none of them tolerates a single packet injection. In 2003, Lysyanskaya et al. [20] designed a technique (called in this paper LTT) resistant to packet loss and data injections using Reed-Solomon codes [35] where the number of signature verifications to be performed per block[2] turns out to be $O(1)$ as a function of the block length n. Unfortunately that approach does not allow lost content to be recovered. This drawback was also present in Tartary and Wang's scheme [41]. This problem was overcome by Karlof et al. in 2004. In [16], they designed a protocol called PRABS using an erasure code (to recover data loss) along with a one-way accumulator [3, 4, 25, 26] based on a Merkle hash tree [23] (to deal with injections). Their approach is similar to Wong and Lam's construction but it has the advantages of allowing recovery of all original data packets with only $O(1)$ signature verifications to compute. Unfortunately PRABS's augmented packets[3] must still carry $\lceil \log_2 n \rceil$ hashes.

In this paper, we propose a scheme having the advantages of the previous two constructions without their drawbacks. Every single original data packet will be recovered by any receiver and, contrary to [20] where only an asymptotic study was performed, we will exhibit an upper bound on the number of signature verifications to be executed per block. This bound will be valid for any block length and will turn to be $O(1)$ as in [16, 20]. Such a bound is valuable for practical applications since the block length is always finite. It allows receivers to get an upper bound on the time spent to verify signatures and therefore on the delay[4] between reception of information and authentication of correct packets. To the best of our knowledge PRABS is the only multicast stream authentication protocol ever designed which exhibits the recovery property and such a complexity value for signature verifications at the same time. As said earlier, this is at the cost of a logarithmic number of hashes appended to each packet. Our

[1] Since the stream size is large, it is divided into small fixed-size entities called *packets*.

[2] In order to be processed, packets are gathered into fixed-size sets called *blocks*.

[3] We call *augmented packets* the elements sent into the network. They generally consist of the original data packets with some redundancy used to prove the authenticity of the element.

[4] This delay is called *authentication delay* of the scheme.