Resource Requirement Analysis for a Predictive-Hashing Based Multicast Authentication Protocol*

Seonho Choi[1] and Yanggon Kim[2]

[1] Department of Computer Science, Bowie State University, 14000 Jericho Park Rd., Bowie, MD 20715, U.S.A.
schoi@bowiestate.edu
[2] Department of Computer & Information Sciences, Towson University, Towson, MD 21252, U.S.A.
ykim@towson.edu

Abstract. A new multicast authentication scheme for real-time streaming applications was proposed [28] that is resistant to denial-of-service attacks with less resource usages (CPU and buffer) at receivers compared to previously proposed schemes. This scheme utilizes prediction hashing (PH) and one-way key chain (OKC) techniques based on erasure codes and distillation codes. Detailed protocol description is presented at the sender and receiver sides, and a worst-case resource (memory and CPU) requirement at the receiver-side is obtained with an assumption of security condition.

Keywords: denial-of-service, multicast, authentication, protocol, resource requirement, cryptographic hashing.

1 Introduction and Related Works

We developed an efficient multicast authentication scheme [28] for real-time streaming applications that is resistant to denial-of-service attacks while consuming less resources (CPU and buffer) at receivers compared to previously proposed schemes. This scheme utilizes *prediction hashing* (PH) and *one-way key chain* (OKC) techniques based on erasure codes [12, 13, 22, 23, 24] and distillation codes [10]. PH and OKC techniques enable the receiver to significantly reduce the CPU overhead and buffer requirements compared to other block-based solution approaches [8, 10, 15, 16, 17, 18]. Our scheme is based on block-based approach where the real-time data stream is divided into blocks of packets and each block includes predictive authentication information for the next block as well as original stream data from the current block. Preliminary analysis conducted in the middle of this paper indicates that this new scheme consumes much less CPU and buffer space than one of the recently proposed denial-of-service (DoS) resistant multicast authentication schemes, pollution resistant authenticated block streams (PRABS) [10], by a factor of more than 5 for buffer requirement and 3 for CPU requirement as will be shown at the end of Section 3.

* This work was supported by US Army Research Office grant 48575-RT-ISP.

X. Zhou et al. (Eds.): EUC Workshops 2006, LNCS 4097, pp. 302 – 311, 2006.

Several solution approaches were proposed for multicast authentication [8, 10, 12, 13, 14, 15, 17, 18, 19, 20, 22, 23, 24, 27]. However, all of the approaches are vulnerable to *denial-of service (DoS)* attacks [10]. Hash graph protocols and the Wong-Lam scheme are vulnerable to signature flooding attacks. An adversary flooding the stream with invalid signatures will overwhelm the computational resources of receivers attempting to verify the signatures. Additionally, in hash graph protocols, adversarial loss patterns can cause denial of service. For example, if an adversary causes the loss of all signature packets, nothing is verifiable. Also, erasure codes based approaches have limitations: erasure codes are designed to handle only a specific threat model: packet loss. Erasure codes assume that symbols are sometimes lost but not corrupted in transit; this is the erasure channel model. Unfortunately, the assumptions that underlie erasure codes are unrealistic in hostile environments. Adversaries can pollute the message stream by injecting invalid symbols. If erasure codes use an invalid symbol as input to its decoding algorithm, it will reconstruct invalid data. We have to assume more realistic attack model: malicious end hosts and routers can observe, inject, modify, delay, and drop messages in an erasure encoded multicast stream [10].

Recently, distillation codes [10] were developed to address DoS attack issues in erasure codes based authentication approach. The new block based authentication protocol, named as *pollution resistant authenticated block streams* (PRABS), was designed and presented in [10] to cope with pollution attacks. However, in PRABS, receivers still need to use significant amount of buffer space, and the CPU overhead at receiver is proportional to the number of attack partitions that may be launched simultaneously.

We developed a new mechanism, which is based on *Prediction Hashing* (PH) and *One-way Key Chain* (OKC), to overcome those limitations. The basic idea of prediction hashing is that each block of packets convey authentication information that will be used to authenticate (or predict) the next block packets instead of sending the authentication information within the same block as in previous approaches [8, 10, 15, 16, 17, 18]. PH technique allows the receivers to save significant amount of buffer space since only the authentication-related portions from each packet needs to be saved for future packet authentication, while the message portions of arrived packets are processed (or authenticated) immediately after each of them is retrieved from the packet buffer. However, in our scheme, the sender side needs to keep the message portions from two consecutive blocks in the buffer to calculate PH.

Explanation on the proposed protocol is given in Section 2 along with attack types considered and the feasibility condition of such attacks. In Section 3, resource requirement analysis is conducted on CPU and memory requirements at the receivers in the worst case scenario. Conclusion is given in Section 4.

2 Predictive Hashing with One-Way Key Chain

We developed a new mechanism, which is based on *Prediction Hashing* (PH) and *One-way Key Chain* (OKC), to significantly reduce resource requirements by the receiver even in the presence of DoS attack packets flowing in. The basic idea of prediction hashing is that each block of packets convey authentication information

that will be used to authenticate (or predict) the next block packets instead of sending the authentication information within the same block as in previous approaches [8, 10, 15, 16, 17, 18]. PH technique allows the receivers to save significant amount of buffer space since only the authentication-related portions from each packet needs to be saved for future packet authentication, while the message portions of arrived packets are processed (or authenticated) immediately after each of them is retrieved from the packet buffer. However, in our scheme, the sender side needs to keep the message portions from two consecutive blocks in the buffer to calculate PH.

One-way key chain technique is already used in other contexts such as in one-time password [11], TESLA [19, 20], etc. In our approach, the sender will obtain a hash chain by applying hash operations recursively to some seed value, and the obtained key values will be assigned to the blocks in backward order of their generation times. The sender will use the assigned key to calculate Message Authentication Codes (MAC) images of the prediction hashes/signature information for the next block, and attach them (along with other authentication related information) to the current block packets. Also, each block packet reveals the key used in the previous block to let the receivers use it in authenticating the previous block packets (or partitions) without applying erasure decoding and signature verifications in most of the cases. These mechanisms are combined with erasure codes and distillation codes to develop a multicast authentication protocol which is very resistant to Denial-of-Service attacks and resource-efficient. Figure 1 shows the overview of our approach at sender side. The receiver side operation is the reverse of the process shown in Figure 1.

At the sender side, erasure codes are applied to the hashes/signature obtained from next block (B_{i+1}) to cope with packet losses during transmission. These erasure encoded symbols are, then, divided into n pieces denoted as $E_1, E_2,\ldots, E_n$. Message authentication codes (MAC) are applied to these E_1 through E_n with a key K_i to prevent a most sophisticated DoS attack which is named as *strong relay-attack*[1] in this paper. Note that the key used in this process will not be revealed until the next block packet is sent out. The output from these MAC codes are denoted as $\Gamma_1, \Gamma_2, \ldots, \Gamma_n$. These $<E_j, \Gamma_j>$ pairs attached to the packet are protected by attaching D_j obtained from the distillation codes. These D_js will allow the receiver to formulate candidate partitions by applying one-way accumulators based on Merkel hash trees. When a certain *security condition* (which will be introduced later) is met, the receiver doesn't have to apply erasure decoding and signature verification operations, which are the most expensive operations in terms of CPU overhead, for each *invalid* partition. Applying only some hash operations will allow the receiver to filter out invalid partitions – due to prediction hashes and one-way key chain techniques. The sender side may choose a proper value of block period, p, to satisfy the security condition. It will be shown that this condition is general enough in most of the cases as long as the real-time constraints are not very tight. Even when the security condition may not be satisfied for every case and receiver, it will be much more difficult to launch effective DoS attacks in our scheme due to the restrictions imposed by PH and OKC techniques on adversaries.

[1] This will be formally defined later in this section.

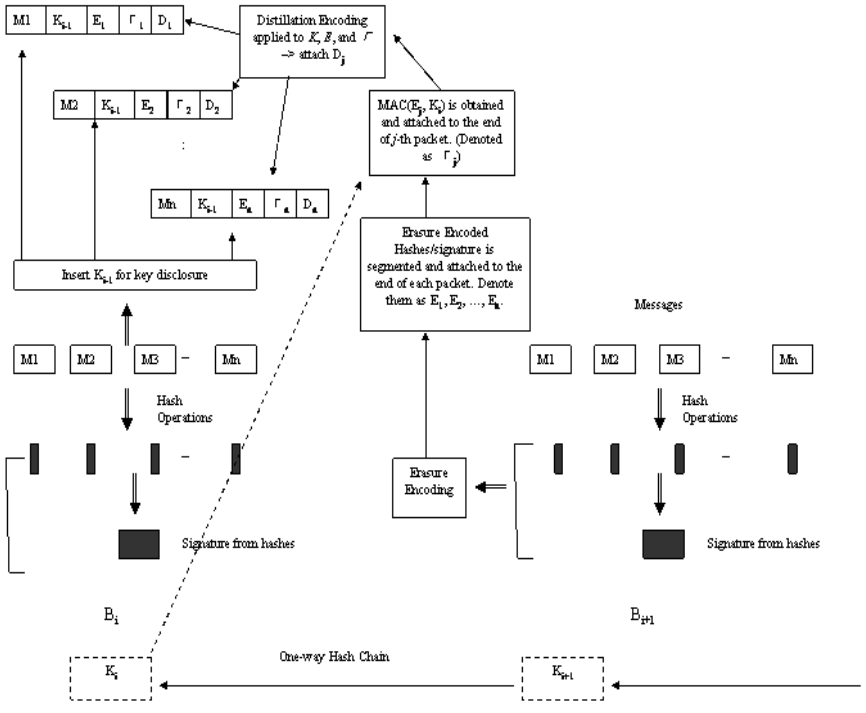

Fig. 1. Overview of our scheme with PH and OKC at the sender

For the purpose of analysis, the following attack types are introduced and used in this paper.

Relay-Attack: An adversary may eavesdrop authentic packets and spoof (and send) packets with invalid authentication-related attachments (while preserving the authentic message portions in each packet) in such a way that the receiver will receive at least *n-t* spoofed packets earlier than *n-t* authentic packets in the same block.

Strong Relay-Attack: If an adversary has the following capabilities, he/she can launch strong relay-attacks:

- Adversary can eavesdrop at least *n-t* authentic packets in B_i and at least one authentic packet in B_{i+1}.
- Adversary copies authentic message portions from B_i packets into (at least) *n-t* spoofed packets, and uses a disclosed key K_i (from B_{i+1} packet) to come up with modified E_j and Γ_j in each spoofed packet.
- Adversary sends all these at least *n-t* spoofed packets in such a way that at least *n-t* of them will be received before a receiver receives *n-t* authentic packets in B_i.

2.1 Feasibility of Strong Relay Attacks

Security condition can be obtained under which the adversaries cannot launch strong relay-attacks. Once the system parameters such as block period (p), block size (n), redundancy level (related to t), etc., are chosen to satisfy this condition, then such

attacks may not be launched by any adversary, thus maximizing the efficiency of our scheme in terms of buffer and CPU usages. Figure 2 shows a diagram for obtaining such condition. If we set the period, p, to be larger than $\delta = d + (n\text{-}t\text{-}1)\ (p/n)$ where d represents the maximum delay from the sender to the receiver, then it would not be possible for any attacker to launch strong relay-attacks.

$$p > d \cdot n/(t+1) \tag{1}$$

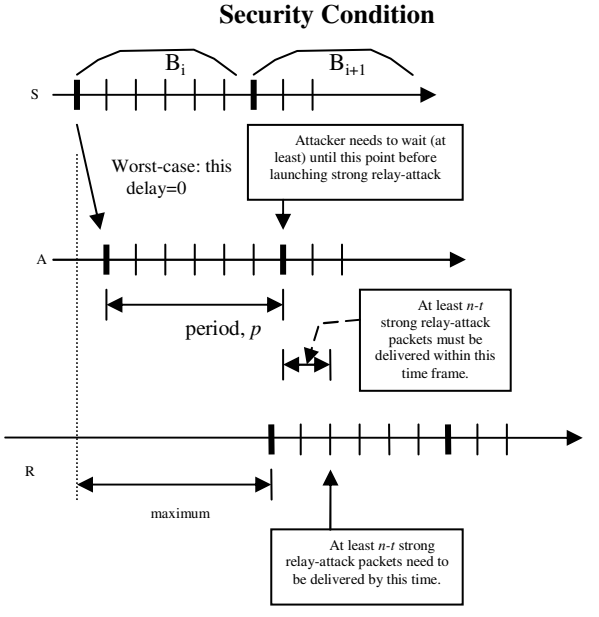

Fig. 2. Security Condition

3 Resource Requirement Analysis

3.1 CPU Overhead

We will estimate the CPU overhead in terms of how many erasure decoding, signature verification, and hash operations are needed in each block for our extended scheme. If we assume that a safe period value is chosen from formula (2) in such a way that no strong relay-attacks may be launched, then the CPU overhead may be specified as follows:

- Number of erasure decoding and signature verification operations: Only one erasure decoding and one signature verification operations are needed for each block. This is because, in our algorithm (Figure 3) – step (4), only those partitions in SymbolBuffer which have Γ_j matching to E_j in a chosen member will be decoded/verified in steps (4-1) and (4-2). If any attacker can't launch strong relay-attacks, any invalid partition stored in SymbolBuffer will not have Γ_j matching to E_j in all its members.

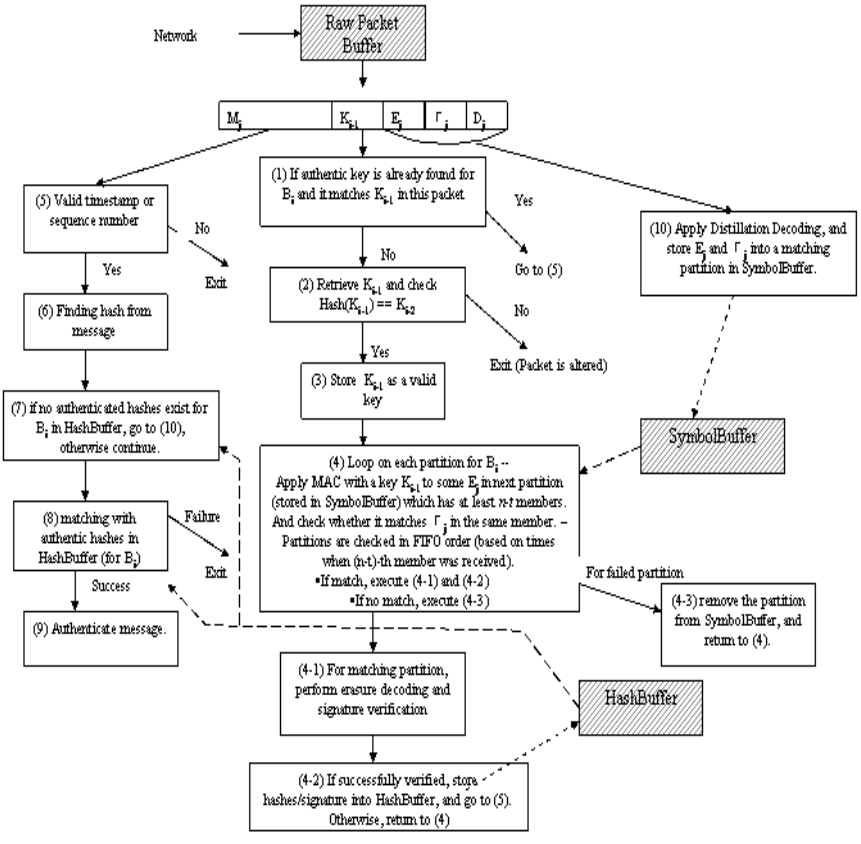

Fig. 3. Detailed Algorithm at a Receiver

- One case we need to consider is when the receiver couldn't obtain any authentic packet in B_i, and, as a result, no hashes/signature information will be stored in SymbolBuffer. And, when the authentic packets in B_{i+1} arrive later, the receiver can't authenticate message portions, however, the hashes/signature information contained in B_{i+1} packets will be stored in SymbolBuffer for B_{i+2} message authentication. Also, it needs to check $Hash(Hash(K_i)) == K_{i-2}$ since the receiver will not have the authentic key K_{i-1}. Even when this happens, the receiver still needs to carry out only one erasure decoding and one signature verification operation when the first authentic packet in B_{i+2} is received.

▪ Number of cryptographic hash operations: Hash operations are applied in steps, (2), (4), (6), and (10).

- At step (2), the worst case occurs when all the attack packets have different keys included except for authentic packets. Note that these hash operations are applied even before Distillation Decoding is applied. In this worst case, the number of hash operations needed in one block is *1*(for authentic partition)+*fn* (for attack packets) = *1*+*fn*. Another extreme case is when all the attack partitions are of size

n-t packets – this case triggers the highest CPU overhead in PRABS. In this case the number of hash operations is equal to the number of partitions with at least *n-t* members, which is *floor(fn/(n-t))*. In other cases, the number of hash operations is in between these two extremes.

- At step (4), the maximum number of hash operations needed in one block is equal to *1+floor(fn/(n-t))* since only one hash operation is required for each attack partition which has at least *n-t* members arrived.
- At step (6), the number of hash operations needed in one block is *(f+1)n* in the worst case when all the attack packets are launched from relay-attacks and have valid key included in them (but, with invalid E_j and Γ_j). Also, message portions form authentic packets need to be hashed, too.
- At step (10), the number of hash operations needed in one block is $(f+1)n \cdot \log n$ in the worst case when attack partitions have *n-t* members each and all the attack packets are launched from relay-attacks and have valid message portions and valid keys included in them (but, with invalid E_j and Γ_j). Also, distillation decoding needs to be performed for authentic packets, too.
- Hence, the total number of hash operations in the worst case is:

$$1+fn+1+ floor(fn/(n-t)) + (f+1)n +(f+1)n \cdot \log n = 2+(2f+1)n + floor(fn/(n-t)) + (f+1)n \cdot \log n \quad (2)$$

From this formula, if we subtract the number of hash operations in Prediction Hash based approach (or PRABS), then we get $2+fn+floor(fn/(n-t)) \approx O(fn)$.

- The total CPU processing time may be represented as follows:

$$[2+(2f+1)n + floor(fn/(n-t)) + (f+1)n \cdot \log n] \cdot C_H + C_E + C_S. \quad (3)$$

3.2 Buffer Requirement

There are three different buffer spaces maintained by the receiver, Raw Packet Buffer, SymbolBuffer, and HashBuffer. Again, for the purpose of simplicity, it is assumed that strong relay-attacks cannot be launched by any attacker. Under this simplified assumption, let's find out the amount of buffer spaces needed:

- Raw Packet Buffer: The worst case scenario occurs when the receiver receives the first authentic packet in each block. In this case, all the steps in the algorithm will be executed from step (1) to (10), including one erasure decoding, one signature verification, and $1+\log n$ hash operations (for steps (7) and (10)). The other cases which require the second longest processing times are when the steps (5) though (10) are executed with steps (2) through (4) are skipped. In this case, at most $1+\log n$ hash operations are needed for processing each packet (steps (7) and (10)). Hence, the total buffer space needed for storing raw packets is:

$$M[(f+1)R \times (C_E + C_S + C_D + (1+\log n)C_H) - 1] \quad (4)$$

- SymbolBuffer: The maximum space is needed when the receiver receives the first authentic packet in the current block with the smallest delivery delay from the sender, and the last authentic packet in the next block at its latest time instant with the largest delivery delay. This is because even when only one authentic packet (e.g., last one) in the next block arrives, the hashes/signature may have to be

extracted and verified from the SymbolBuffer to verify the authenticity of the received packet.

$$r_2 \cdot (2p + d)(f+1) \quad (5)$$

Here, d is the maximum delay that can be experienced by the receiver. And, r_2 is the byte arrival rate (bytes/sec) for E_j and Γ_j that need to be stored in SymbolBuffer.

- HashBuffer: This buffer is needed for storing authentic hashes/signature once they are authenticated at step (4-2). One set of authentic hashes/signature will no longer be required when the possibility of packet arrivals for the next block is none. Also, while the hashes/signature in HashBuffer are being used for verifying the next block packets, their attached portions (in the next block packets) may be erasure decoded and verified. Hence, double buffering type of technique needs to be used for storing hashes/signature for two consecutive blocks at any time. Thus, the buffer requirement is:

$$2r_2 \cdot p(n-t)/n \quad (6)$$

- Total Buffer space needed is:

$$M[(f+1)R \times (C_E + C_S + (1+\log n)C_H) - 1] + r_2 \cdot (2p + d)(f+1) + 2r_2 \cdot p(n-t)/n \quad (7)$$

Table 1. Buffer and Per-Block CPU requirements for Prediction Hash based approach (including Prediction Hash with One-way Key Chain) and PRABS

	Buffer Requirement	(Average) CPU requirement
PRABS	$(r_1+r_2+r_3)(p+d)(f+1) +$ $M[(f+1)R \times (f+1)(C_E + C_S + C_D) - (f+1)]$	$(n + n \times \log n)(f+1)C_H +$ $(1+fn/(n-t))(C_E+C_S)$
Predictiive Hash	$r_2 \cdot (2p + d)(f+1) + 2r_2 \cdot p(n-t)/n +$ $M[(f+1)R \times (C_E + C_S + (1+\log n)C_H) - 1]$	$[2+(2f+1)n + floor(fn/(n-t)) + (f+1)n \cdot \log n] \cdot C_H + C_E + C_S$

Example figures are obtained from the following realistic numbers [12, 29] and the results are shown in Table 2.

$p = 0.3$, $d = 0.1$, $f=10$, $n=8$, $t=4$, $M=500$, $r_1=3120$ bytes/sec, $r_2=400$ bytes/sec, $r_3=480$ bytes/sec, $C_E=0.005$, $C_S=0.005$, $C_H=0.000125$.

One hash is assumed to occupy 20 bytes. It is shown that our scheme with PH and OKC improves the performance of the previous scheme with just PH technique, and our scheme outperforms PRABS with significant improvements.

Table 2. Buffer and Per-Block CPU requirements for an example system

	Worst-case Buffer Requirement	(Worst-case) Per-block CPU time
PRABS	*69905 bytes*	*0.254 seconds*
Prediction Hash with One-way Key Chain	*11706 bytes*	*0.067 seconds*

4 Conclusion

We developed a new authentication scheme based on PH and OKC techniques on top of erasure codes and distillation codes to provide enhanced resistance to DoS attacks while consuming much less resources compared to other block-based multicast stream authentication schemes. We also analyzed the worst-case resource requirements under the assumption that the security condition is satisfied, and found out that much less resources are needed compared to other protocols such as PRABS.

References

1. D. Adkins, K. Lakshminarayanan, A. Perrig, and I. Stoica. Taming IP packet flooding attacks. In *Proceedings of Workshop on Hot Topics in Networks (HotNets-II)*, Nov. 2003.
2. T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet denial-of-service with capabilities. In *Proceedings of Workshop on Hot Topics in Networks (HotNets-II)*, Nov. 2003.
3. N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology --EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494, 1997.
4. M. Bellare and P. Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 470–484, 1997.
5. J. Benaloh and M. de Mare. One way accumulators: A decentralized alternative to digital signatures. In *Advances in Cryptology – EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285, 1993.
6. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology – CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76, 2002.
7. V. Gligor. Guaranteeing access in spite of service-flooding attacks. In *Proceedings of the Security Protocols Workshop*, Apr. 2003.
8. P. Golle and N. Modadugu. Authenticating streamed data in the presence of random packet loss. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS 2001)*, pages 13–22. Internet Society, Feb. 2001.
9. M. Goodrich, R. Tamassia, and J. Hasic. An efficient dynamic and distributed cryptographic accumulator. In *Proceedings of Information Security Conference (ISC 2002)*, volume 2433 of *Lecture Notes in Computer Science*, pages 372–388, 2002.
10. C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. Tygar, Distillation codes and applications to DoS resistant multicast authentication, in Proc. 11th Network and Distributed Systems Security Symposium (NDSS), San Diego, CA, Feb. 2004.
11. Leslie Lamport, "*Password Authentication with Insecure Communication*", Communications of the ACM 24.11 (November 1981), 770-772
12. M. Luby. LT codes. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '02)*, pages 271–282, 2002.
13. M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann. Practical loss-resilient codes. In *Proceedings of 29th Annual ACM Symposium on Theory of Computing (STOC '97)*, pages 150–159, May 1997.
14. R. Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 122–134, Apr. 1980.

15. S. Miner and J. Staddon. Graph-based authentication of digital streams. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 232–246, May 2001.
16. A. Pannetrat and R. Molva. Efficient multicast packet authentication. In *Proceedings of the Symposium on Network and Distributed System Security Symposium (NDSS 2003)*. Internet Society, Feb. 2003.
17. J. M. Park, E. Chong, and H. J. Siegel. Efficient multicast packet authentication using erasure codes. *ACM Transactions on Information and System Security (TISSEC)*, 6(2):258–285, May 2003.
18. J. M. Park, E. K. Chong, and H. J. Siegel. Efficient multicast packet authentication using signature amortization. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 227–240, May 2002.
19. A. Perrig, R. Canetti, D. Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS 2001)*, pages 35–46. Internet Society, Feb. 2001.
20. A. Perrig, R. Canetti, J. D. Tygar, and D. Song. Efficient authentication and signature of multicast streams over lossy channels. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 56–73, May 2000.
21. A. Perrig and J. D. Tygar. *Secure Broadcast Communication in Wired and Wireless Networks*. Kluwer Academic Publishers, 2002.
22. M. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348, 1989.
23. I. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.
24. L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, 27(2):24–36, Apr. 1997.
25. T. Sander. Efficient accumulators without trapdoor extended abstracts. In *Information and Communication Security, Second International Conference – ICICS '99*, volume 1726 of *Lecture Notes in Computer Science*, pages 252–262, 1999.
26. D. Song, D. Zuckerman, and J. D. Tygar. Expander graphs for digital stream authentication and robust overlay networks. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 258–270, May 2002.
27. C. Wong and S. Lam. Digital signatures for flows and multicasts. In *Proceedings on the 6th International Conference on Network Protocols (ICNP '98)*, pages 198–209. IEEE, October 1998.
28. Seonho Choi, "Denial-of-Service Resistant Multicast Authentication Protocol with Prediction Hashing and One-way Key Chain," ism, pp. 701- 706, In Proceedings of the Seventh IEEE International Symposium on Multimedia (ISM'05), 2005.