

# Efficient Multicast Packet Authentication Using Signature Amortization

Jung Min Park<sup>†</sup>

<sup>†</sup>CERIAS

<sup>†</sup>School of Electrical and Computer Engineering  
Purdue University  
West Lafayette, IN 47907-1285 USA  
parkjm@ecn.purdue.edu

Edwin K. P. Chong<sup>‡§</sup> and Howard Jay Siegel<sup>†\*</sup>

<sup>‡</sup>Department of Electrical and Computer Engineering

<sup>§</sup>Department of Mathematics

<sup>\*</sup>Department of Computer Science

Colorado State University

Fort Collins, CO 80523-1373 USA

{echong, hj}@colostate.edu

## Abstract

*We describe a novel method for authenticating multicast packets that is robust against packet loss. Our main focus is to minimize the size of the communication overhead required to authenticate the packets. Our approach is to encode the hash values and the signatures with Rabin's Information Dispersal Algorithm (IDA) to construct an authentication scheme that amortizes a single signature operation over multiple packets. This strategy is especially efficient in terms of space overhead, because just the essential elements needed for authentication (i.e., one hash per packet and one signature per group of packets) are used in conjunction with an erasure code that is space optimal. To evaluate the performance of our scheme, we compare our technique with four other previously proposed schemes using analytical and empirical results. Two different bursty loss models are considered in the analyses.*

## 1. Introduction

Fueled by the explosive growth of the Internet and growing demand for novel types of group communications, multicast has received a lot of attention in recent years. In multicast, a single copy of packets is sent by the sender and routed to every receiver within the multicast group via multicast-enabled routers. For a wide range of applications, multicast is an efficient and natural way of communicating information. Some examples include information broadcasts (e.g., news feeds, weather updates, and stock quotes), multiparty videoconferencing,

and software updates. For successful implementation, many of these applications will require varying degrees of security requirements (i.e., confidentiality and authentication).

Confidentiality for multicast transmissions can be provided using techniques that utilize symmetric (secret) key cryptography. Confidentiality would be provided by encrypting the message with the secret key being shared by the sender and the receivers of the multicast group before transmission. Consequently, off-the-shelf solutions such as the Advanced Encryption Standard (AES) can be readily employed for this purpose. For confidentiality, the main concern is the complexity involved in key management (e.g., key distribution, revocation, and group updates).

The solution to the authentication problem for unicast transmission is rather simple and well known (e.g., the use of Hash based Message Authentication Codes (HMAC)). However, this solution is inadequate for the multicast setting. The difficulty of the problem lies in the fact that preventing the forgery of packets by a colluding group of receivers precludes the use of any symmetric key cryptosystem that is efficient in terms of space overhead. Without using symmetric key cryptosystems, the most obvious way of providing authentication is to sign each individual packet using the sender's digital signature. However, the computation overhead of current signature schemes is too high to make this practical. According to [17], a Pentium II 300 MHz machine devoting all of its processor time can only generate 80 512-bit RSA signatures and verify 2000 signatures per second. This signature operation would also require 64 bytes of communication overhead per packet. Clearly, this approach is not practical.

Packet loss is another important issue that needs to be considered. While this may not be a problem for applications employing reliable transport protocols (e.g., TCP/IP), it is a serious issue for multicast applications

---

This research was supported in part by the Center for Education and Research in Information Assurance and Security (CERIAS), by the Colorado State University George T. Abell Endowment, and by NSF under grants 0098089-ECS and 0099137-ANI.

that use UDP over IP-Multicast. Because UDP only provides best-effort service, when UDP packets are sent across multiple administrative boundaries with diverse routing topologies and conditions, packet loss can be high. Therefore, while the content being broadcast might be able to bear packet losses, the authenticity of it might not be verifiable by the receiver, if the authentication scheme is not resistant to loss. Current techniques for reliable multicast, such as Scalable Reliable Multicast (SRM) [4] and Reliable Adaptive Multicast Protocol (RAMP) [7], are complex and not yet standardized. Considering the lack of any standardized techniques for reliable multicast transmission, any practical multicast authentication scheme should be robust against loss.

Our approach to multicast message authentication is based on the technique of signature amortization, i.e., amortizing a single signing operation over multiple packets. This technique greatly improves signing and verification rates compared to the naïve signature-per-packet approach. To deal with packet loss, we employ an erasure code to encode the authentication information. Our strategy is especially efficient in terms of space overhead, because just the essential elements needed for authentication (i.e., one hash per packet and one signature per block of packets) are used in conjunction with an erasure code that is space optimal. According to our simulation results (see Subsection 5.2), this technique is highly robust against loss and achieves higher authentication probabilities compared with previously proposed schemes (given the same amount of communication overhead).

In the next section, we give a brief overview of the previous work done in multicast message authentication and a brief overview of erasure codes. The rationale for our approach, along with the detailed authentication/verification procedure, is given in Section 3. In Section 4, we discuss the authentication probability of our scheme using two different bursty loss models. For one of the loss models, we derive the asymptotic authentication probability, and for the other loss model, we give the lower bound. In Section 5, we evaluate the performance of our technique, comparing it with four other previously proposed schemes. Finally, concluding remarks are given in Section 6.

## 2. Background information

### 2.1. Related work

Multicast authentication is an active area of research, and researchers have suggested many schemes, which can be divided into two major classes—computationally secure authentication and unconditionally secure authentication. Pioneering research on unconditionally secure authentication was done by Simmons [15] and later

extended to the multicast setting by Desmedt et al. [3]. As the name suggests, unconditionally secure authentication provides very strong security guarantees, but is less practical compared to the computationally secure techniques. Our focus will be on the computationally secure methods.

In the realm of computationally secure multicast authentication, two approaches can be taken. One approach is to use MACs based on symmetric key cryptography, and the other approach is to utilize digital signatures based on asymmetric key cryptography. Schemes based on MACs do not use any computationally intensive asymmetric key techniques. A variation of this idea was proposed in [2]. The basic idea is to use  $l$  different keys to authenticate every message with  $l$  different MACs. The sender knows a set of  $l$  keys,  $R = \{K_1, \dots, K_l\}$ , and attaches to each packet  $l$  MACs—each MAC computed with a different key. The message is transmitted together with the concatenated string of MAC values. Each recipient  $u$  knows a subset of the keys  $R_u \subset R$ , where every key  $K_i$  is included in  $R_u$  with probability  $1/(w+1)$ , independently for every  $i$  and  $u$ .

Using these keys, each recipient computes  $|R_u|$  MAC values for each packet. If all of the  $|R_u|$  values match the corresponding MAC values transmitted by the sender, then the packet is verified as being authentic. Appropriate choice of subsets  $R_u$  insures that with high probability no coalition of up to  $w$  malicious users have knowledge of all the keys kept by another user within the multicast group. This technique is applicable in situations where multicast groups are small and collusions can be controlled.

In [11], Perrig et al. propose a scheme based on MACs that requires time synchronization between the sender and the receiver. In their solution, *Timed Efficient Stream Loss-tolerant Authentication (TESLA)*, a MAC is embedded in each packet to provide authentication. The corresponding MAC keys are disclosed to the receiver after a time delay. The delay before the disclosure is chosen long enough so that they cannot be used to forge packets. Although this scheme is robust against packet loss and scalable, it requires that the sender and receiver synchronize their clocks within a certain margin. In settings where time synchronization is difficult to achieve, TESLA might not work.

Boneh et al. [1], in their recent work, showed that one cannot build an efficient (in terms of communication overhead) collusion resistant multicast authentication scheme without relying on digital signatures. Many schemes based on asymmetric key cryptography attempt to reduce the computation and communication overhead by amortizing a single signature over multiple packets.

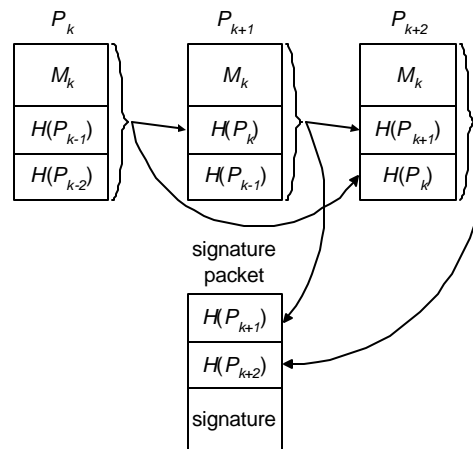
Even if the computational load required for the signature generation is amortized, the communication overhead can be significant if one were to make each packet carry its own authentication information and thus make it individually verifiable. In this kind of a scheme, any packet that is received can be verified. This is the approach taken by Wong and Lam [17]. They employ Merkle's *authentication trees* [9] to reduce the size of the authentication information and sign multicast packets. The underlying idea is to divide a stream into blocks and amortize a single signing operation over a block of packets. Before the signing operation, an authentication tree is computed for each block. In an authentication tree:

- Packet digests (or hashes) are the leaf nodes.
- Other nodes of the tree are computed as message digests of their children.
- The root is the block digest, with the block signature being the signature of the root.

Within the blocks, each packet carries its own authentication information consisting of the signed block digest, the packet position in the block, and the siblings of each node in the path of the packet's corresponding leaf node to the root. To verify a packet, the receiver needs to verify the packet's path to the root and compare the computed block signature with the received one. This technique improves signing and verification rates by an order of magnitude compared to the naïve signature-per-packet approach. Although this method has its own merits, it also has practical limitations. Because it is individually verifiable, each packet needs to contain a signature with all the nodes necessary to compute the root signature, which requires a large communication overhead. In practice, this scheme adds over 200 bytes to each packet (assuming a 1024-bit RSA signature and a block size of 16 packets).

If the condition on individual packet verification is relaxed so that the verification of a packet is dependent on other packets within the block, then the communication overhead can be reduced substantially. In this type of an approach, verification of each packet is not guaranteed and instead is assured with a certain probability. Perrig et al. [11] take this probabilistic approach and use a combination of hash functions and digital signatures to authenticate packets. Their scheme, *EMSS (Efficient Multi-chained Stream Signature)*, is an extension of Gennaro and Rohatgi's *stream signing* technique [5]. The basic idea is as follows: A hash of packet  $P_1$  is appended to packet  $P_2$ , whose hash is in turn appended to  $P_3$ . If a *signature packet*, containing the hash of the final data packet (i.e.,  $P_3$ ) along with a signature, is sent after  $P_3$ , then non-repudiation is achieved for all three packets. In essence, hash values act as chains between the packets so that they form a single string that can be signed by one digital signature. However, this basic approach is not

robust against packet loss—even a single packet loss would break the chain, which would make it impossible to verify the authenticity of the packets preceding the break point. EMSS overcomes this drawback by storing the hash of each packet in multiple locations and appending multiple hashes in the signature packet. For example, each packet  $P_k$  would include hashes  $H(P_{k-1}), H(P_{k-2}), \dots$  of the previous packets  $P_{k-1}, P_{k-2}, \dots$ . The signature packet, which contains the hashes of the final few packets along with a signature, is sent at the end of the stream to authenticate all the packets. Tolerance to loss can be increased further by sending multiple copies of a signature packet—copies would be sent with delayed intervals, because packet loss is correlated. To reduce the verification delay at the receiver side, a stream of packets is divided into blocks, and the same process is repeated for every block, i.e., all the data packets within the block are chained with multiple hashes followed by an insertion of one or more signature packets.



**Figure 1.** An example of EMSS.

A simplified example is shown in Figure 1, where each packet contains hashes of the previous two packets, and the signature packet contains hash values of the final two packets along with a signature. In the figure, a directed edge from  $P_i$  to  $P_j$  indicates that the hash of packet  $P_i$  has been appended to packet  $P_j$ . The authors suggest using random edge distributions to improve performance, especially when packet losses are correlated. EMSS tolerates packet loss with relatively low overhead at the cost of delayed verification. However, as we will see in Section 5.2, to maintain a high *verification probability* in a high-loss environment, its communication overhead must be increased considerably. Throughout the paper, we will use the term *verification probability* interchangeably with *fraction of verifiable packets* to

denote the number of verifiable packets of the stream divided by the number of received packets of the stream.

In [6], Golle and Modadugu propose a similar scheme called the *augmented chain* technique. They propose a systematic method of inserting hashes in strategic locations so that the chain of packets formed by the hashes will be optimally resistant to bursty packet loss, given certain constraints. There are two basic differences between this scheme and EMSS:

- In the augmented chain technique, each packet includes the hashes of the subsequent packets as well as the previous packets (i.e., leftward and rightward edges).
- EMSS stores the hashes in random locations, while augmented chain chooses these locations in a deterministic way.

The chain of packets constructed by the augmented chain method is parameterized by the integer variables  $a$  and  $p$ . The augmented chain is constructed in two phases. In the first phase, a chain is formed among a subset of the packets as follows: the hash of packet  $P_i$  is appended to two other packets  $P_{i+1}$  and  $P_{i+a}$ . In the second phase, the rest of the packets are inserted. Specifically,  $p-1$  additional packets are inserted between each pair of consecutive packets of the original chain (constructed in the first phase) and connected in a systematic way using directed edges. The authors propose two ways of inserting new packets, which are equally robust to packet loss. A chain constructed in this manner can sustain a burst loss of up to  $p(a-1)$  packets in length. To reduce the verification delay, a stream is divided into blocks of packets, and each block is constructed using the augmented chain technique with only the last packet in the block being signed. The interested reader should refer to [6] for further details.

A simple example of an augmented chain is illustrated in Figure 2. In the figure, packets of the original chain are represented by letters, and the newly inserted packets are represented by numbers. According to our simulation results in Section 5, the augmented chain technique achieves higher verification probabilities compared to EMSS (with the same communication overhead) at the cost of increased delay on the sender side. Compared with our authentication scheme, however, augmented chain is less robust against loss given the same amount of communication overhead.

In [10], the authors propose a similar authentication scheme based on hash chaining techniques that is specifically designed to resist multiple bursts within a block. In the construction of their scheme (called the *piggybacking* scheme),  $n$  packets are partitioned into  $r$  equal-sized priority classes, and hence each class is of size  $z = n/r$ . Each packet is represented by  $s_{ij}$ , where  $i$  denotes the class number and  $j$  denotes the packet index

within that class. The signature packet is the first packet in the highest priority class and thus represented by  $s_{11}$ . It is assumed that the packets in the highest priority class are spaced evenly throughout the block, so that, two consecutive packets of the highest priority class are located exactly  $r$  packets apart. By constructing hash chains in the following manner, packet  $P_i$  can tolerate  $x_i$  bursts of size at most  $b_i = k_i r$ :

#### Piggybacking scheme construction:

For  $i = 2, 3, \dots, r$  and  $j = x_i k_i + 1, x_i k_i + 2, \dots, z$ , add edges  $\vec{e}(s_{i,j}, s_{0,j}), \vec{e}(s_{i,j}, s_{0,j-k_i}), \dots, \vec{e}(s_{i,j}, s_{0,j-x_i k_i})$ , where  $\vec{e}(P_i, P_j)$  denotes a hash chain formed by placing a hash of  $P_i$  in  $P_j$ .

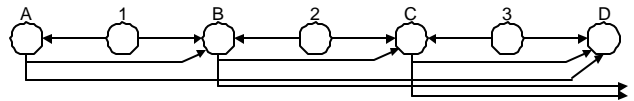


Figure 2. Augmented chain with  $a = 3$ ,  $p = 2$ .

## 2.2. Erasure codes

When a stream of packets are sent from the source to the destination via the Internet, fraction of the packets are lost during transit. A standard solution to this problem is to request retransmission of data that is not received. In some applications, this solution is not practical. For real-time data, this can lead to unacceptable delays. In multicast transmission, different receivers lose different sets of packets, thus retransmissions can overburden the resources of the sender and the network.

An alternative solution is to use forward error correction (FEC) techniques. Using this method, robustness to loss can be provided without imposing an unreasonable amount of space overhead. Among FEC codes, *erasure codes* are of particular interest to our application. In this subsection, we briefly review the basic characteristics of two well-known erasure codes—Rabin’s *Information Dispersal Algorithm (IDA)* [12] and *Tornado codes* [8].

IDA was originally developed as an algorithm for providing safe and reliable storage or transmission of information in distributed systems. The basic idea of IDA is to process the source, say a file  $F$ , by introducing some amount of redundancy and splitting the result into  $n$  pieces, which are then transmitted. Reconstruction (decoding) of  $F$  is possible with any combination of  $m$  pieces (assuming  $n-m$  pieces are lost during transmission), where  $m \leq n$ . Each distributed piece is of size  $|F|/m$ , which clearly shows that the scheme is space optimal. The space overhead for transmission can be controlled by adjusting the parameters  $n$  and  $m$ .

Unlike IDA, which has a quadratic decoding time, Tornado codes can encode and decode data in linear time. The number of segments needed for decoding is slightly larger than the number of pre-encoded segments, and thus they are sub-optimal in terms of space overhead. Specifically, for a set of  $n$  segments, encoding with Tornado codes increase the number to  $n/\{1-p(1+e)\}$ , where  $p$  and  $e$  are positive fractions. If the receiver acquires more than  $1-p$  fraction of the encoded segments, then the original data segments can be reconstructed with high probability in time proportional to  $n \ln(1/e)$ .

### 3. Our scheme for message authentication

#### 3.1. Rationale for our approach

In EMSS, there are three factors that affect verification probability—number of signature packets, number of hashes contained in the signature packet, and number of hashes contained in the data packet. The number of hashes in the signature and data packets (i.e., second and third factors) is always greater than one, improving the robustness of the scheme to packet loss compared to the single hash-per-packet configuration of the stream-signing technique [5]. This tradeoff of increased space overhead for packet loss robustness is unavoidable, but can be done in a more efficient way by using erasure codes. This is best illustrated using the following simple example.

The sender transmits the hash of a packet appended to  $k$  other packets for increased resistance to loss. We assume that a block consists of  $n$  packets. If independent packet loss is assumed, then the probability that at least one out of the  $k$  packets will reach the destination is  $1-q^k$ , where  $q$  is the packet loss probability. The communication overhead would be  $kh$ , where  $h$  is the size of the hash. Using the same overhead, one can encode the hash using IDA and append the encoded  $n$  segments to the  $n$  packets of the block (i.e., each packet in the block would contain one of the encoded segments). The minimum number of encoded segments needed for reconstruction of the hash is only  $m = \lceil n/k \rceil$ , where  $\lceil x \rceil$  denotes the smallest integer not less than  $x$ . The probability that the hash can be reconstructed successfully at the receiver is given by

$$1 - \sum_{i=0}^{m-1} \binom{n}{i} (1-q)^i q^{n-i}. \quad (1)$$

Again, independent packet loss was assumed.

Instead of using IDA, the sender could also use one of the probabilistic codes such as Tornado codes for the same purpose. In this case, the minimum number of segments needed for decoding is

$$m = n(1-p) = \left\lceil \frac{n}{k} \left( 1 + \frac{e(k-1)}{e+1} \right) \right\rceil, \quad (2)$$

where  $p$  and  $e$  are the performance parameters of the Tornado code (see Subsection 2.2). The probability of successful reconstruction of the hash is given by (1) using the value of  $m$  specified by (2). It is obvious that the probability given by (1) is much higher than  $1-q^k$ , and the probability for IDA is higher than that for the Tornado code.

The above example suggests that using some type of an erasure code to encode the hash values would be more efficient than simply appending duplicate hash values to the packets. As an extended version of EMSS, Perrig et al. [11] suggested using universal hash functions or IDA to split the hash value of each packet into multiple pieces before appending them onto other packets. This certainly produces a more loss-resistant scheme with the same amount of communication overhead. However, it introduces complexities—the time-consuming process of encoding and decoding must be performed for each hash. This can be a bottleneck, especially when one of the erasure codes from the Reed-Solomon family, which has quadratic decoding times, are employed. We suggest a simple method of avoiding this problem at the cost of sender delay. Instead of encoding individual hashes, we suggest concatenating all the hashes within the block to form a single piece before encoding. This strategy requires only one encoding/decoding per block. The space overhead can be minimized by employing space optimal erasure codes such as IDA.

The main advantage of EMSS is that there is no sender delay incurred by the authentication process—a given packet can be transmitted immediately after its hash value is computed without the buffering of other packets. This can be an advantage in situations where data is generated in real time, and immediate dissemination is crucial (e.g., stock-quote broadcasts). However, for most multicast applications, the sender has *a priori* knowledge of at least a portion of the data (e.g., pre-recorded news footage and software updates). In fact, most authentication schemes incur some degree of sender delay to gain other advantages [6, 10, 17]. Therefore, the sender delay caused by our technique is not a restrictive requirement in most multicast applications.

If a certain amount of sender delay is allowed, then a more significant problem can be addressed. It is obvious that the delivery of the signature packets is crucial for any authentication scheme. In previous work [6, 10, 11], the performance results (both analytical and empirical) were based on the assumption that the signature packet is received. The authors suggest accomplishing this task by empowering the receivers to request retransmissions of

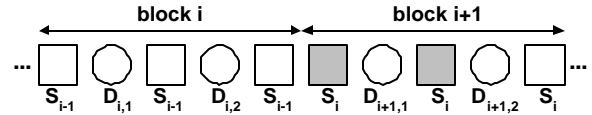
the lost signature packets or sending the signature packet multiple times. However, the retransmission of signature packets can put considerable strain on the resources of the sender and the network, especially in a large multicast network consisting of tens of thousands of users. In [18], Yajnik et al. observed packet loss characteristics of actual multicast sessions, and showed that considerable amounts of the packets would need to be retransmitted, if reliable multicast services are to be provided through retransmissions. In one particular data set, 62.6% of the packets sent by the source were lost by at least one receiver. This implies that retransmission would have been necessary for 62.6% of the packets.

Sending the signature packet several times can be an alternate solution, but this also has drawbacks. Signature packets are generally large (e.g., 128 bytes, if 1024-bit RSA is used) and sending these packets several times can increase the communication overhead noticeably. There is another drawback to sending the signature packet multiple times. Because actual losses in the Internet are highly correlated and bursty, each copy of the signature packet would have to be interspersed uniformly among the packets to ensure maximum effectiveness. If the copies of the signature packets are distributed in the current block, then this would cause sender delays in schemes that utilize hash chaining techniques with edges directed rightward (i.e., hash of a packet is appended to the packets following it)—schemes like EMSS. The sender delay is caused by the fact that the data packets in the block cannot be transmitted before the signature packet is created, and replicas of it are interspersed among the data packets. In contrast, distributing copies of the signature packets in the current block would not add any additional sender delay in schemes that utilize leftward edges (these schemes already incur some amount of sender delay)—techniques like the piggybacking scheme [10].

The obvious alternative is to distribute the copies in the next block to avoid the sender delay. However this can cause increased delay on the receiver side—a receiver might have to buffer a maximum of  $2i$  data packets before verifying a given packet, where  $n$  is the number of data packets per block. This case is illustrated as a simple example in Figure 3. In the figure, circles and squares represent data packets and signature packets, respectively. The first two signature packets in the  $(i+1)$ -th block are assumed to be lost and are represented as darkened squares. The receiver needs to buffer  $D_{i,1}$ ,  $D_{i,2}$ ,  $D_{i+1,1}$ , and  $D_{i+1,2}$  before verifying the data packets of the  $i$ -th block (i.e.,  $D_{i,1}$  and  $D_{i,2}$ ) using  $S_i$  (signature packet of the  $i$ -th block).

Considering these problems, the obvious alternative is to apply FEC techniques to the signatures packets. We can easily make the signature packets robust against packet loss by using erasure codes and appending each

encoded piece to the data packets. If IDA is used, the same set of vectors used for encoding/decoding the concatenated hash values can be used, and hence no additional (computational) encoding/decoding overhead is added. In addition, there is no additional sender/receiver delay involved with encoding/decoding signature packets. The details of our authentication scheme are given in Section 3.2.



**Figure 3.** Increase in receiver delay caused by multiple signature packet transmissions.

For our authentication scheme, we employed IDA instead of probabilistic codes such as Tornado codes. Tornado codes can encode/decode data very rapidly (i.e., linear time), but do so with a high probability only when the number of segments to encode is large. For this reason, Tornado codes are appropriate when large number (hundreds to thousands) of segments are being encoded [16]. We chose to use IDA, because the encoding involves a fairly small number of segments, and IDA has the added advantage of being space optimal. It should be noted that our authentication scheme is independent of the type of erasure code, and other erasure codes (e.g., Tornado codes) that have other attractive properties can be employed in the scheme.

### 3.2. Signature amortization using IDA

The biggest challenge in using digital signatures for authentication is the computationally intensive nature of the asymmetric-key-based signatures. For this reason, previous authentication schemes approached this problem in two directions—designing faster signature techniques and amortizing the signature operation over multiple packets.

In general, making the signature scheme faster comes at the cost of increased space overhead. In [13], Rohatgi proposes using a combination of  $k$ -time signatures and certificates for the  $k$ -time public keys (created with a regular signature scheme) to authenticate packets. Despite its improvement over the one-time signature scheme, this method still requires a space overhead on the order of several hundred bytes per packet.

Our main focus is reducing the size of the authentication overhead, and therefore we took the second approach, which offers better space efficiency. We propose a scheme that employs IDA to amortize a single digital signature over a block of packets. Our scheme, appropriately named *Signature Amortization using IDA* (*SAIDA*), was designed to provide space-efficient

authentication even in high packet-loss environments. The following steps describe the authentication procedure in detail:

1. Let  $\parallel$  denote concatenation. A stream of packets is first divided into groups (or blocks). We denote a stream as  $\mathcal{G} = G_1 \parallel G_2 \parallel \dots$ , where each group  $G_i$  is a concatenated string of  $n$  packets (i.e.,  $G_i = P_{(i-1)n+1} \parallel \dots \parallel P_{in}$ ), and each packet  $P_i \in \{0, 1\}^c$  for some constant  $c$ . The same operations are performed on every group, so we will only focus on the first group.

2. A *packet hash*  $H(P_i), i=1, \dots, n$  for each packet is computed using a hash function  $H$  such as MD5 or SHA-1.

3. The packet hashes are concatenated to form  $F^1 = H(P_1) \parallel \dots \parallel H(P_n)$  of size  $N$  (i.e.,  $F^1$  consists of  $N$  characters). Let  $b_i$  represent the  $i$ -th character in  $F^1$ . In practice,  $b_i$  may be considered as an eight-bit byte, hence  $0 \leq b_i \leq 255$ , and all computations are done in  $\mathbf{Z}_{257}$  or  $GF(2^8)$ . One copy of  $F^1$  is stored in a temporary buffer while another copy is divided into blocks of length  $m$  as follows:

$$F^1 = (b_1, \dots, b_m), (b_{m+1}, \dots, b_{2m}), \dots, (b_{N-m+1}, \dots, b_N).$$

To simplify the following discussion, let

$$\mathbf{S}_i = (b_{(i-1)m+1}, \dots, b_{im}), 1 \leq i \leq N/m.$$

4. Choose  $n$  vectors  $\mathbf{a}_i = (a_{i1}, \dots, a_{im}), 1 \leq i \leq n$ , such that every subset of  $m$  different vectors are linearly independent, as specified in [12].

5. Using the set of vectors  $\mathbf{a}_i = (a_{i1}, \dots, a_{im}), 1 \leq i \leq n$ ,

$F^1$  is processed and divided into  $n$  pieces as follows:

$$F_i^1 = (\mathbf{a}_i \cdot \mathbf{S}_1, \mathbf{a}_i \cdot \mathbf{S}_2, \dots, \mathbf{a}_i \cdot \mathbf{S}_{N/m}), i = 1, \dots, n,$$

where  $\mathbf{a}_i \cdot \mathbf{S}_k = a_{i1} \cdot b_{(k-1)m+1} + \dots + a_{im} \cdot b_{km}$ .

It follows that  $|F_i^1| = |F^1|/m$ .

6. The *group hash* is computed by taking the hash of the other copy of  $F^1$  as follows:

$$H_G(G_1) = H(F^1) = H(H(P_1) \parallel \dots \parallel H(P_n)),$$

where  $H_G(G_1)$  denotes the group hash of the first group of packets.

7. The group hash is signed by an asymmetric-key signature scheme using the sender's private key  $K_r$  and denoted as  $\mathbf{s}(K_r, H_G(G_1))$ . Once again, this value is processed and divided into  $n$  segments using the same set of vectors and is expressed as

$$\mathbf{s}_1(K_r, H_G(G_1)), \dots, \mathbf{s}_n(K_r, H_G(G_1)).$$

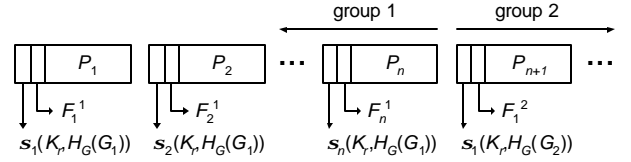
Although, this procedure was explained as a separate step for clarity, the signature can be concatenated with

$F^1$ , before applying IDA, so that only one encoding per group is necessary.

8. Each signature segment (created in the above step) and hash segment (created in the fifth step) are concatenated with the corresponding packet to form an authenticated packet. A group of  $n$  authenticated packets combine to form an authenticated group, which is expressed as

$$\mathbf{S}_1(K_r, H_G(G_1)) \parallel F_1^1 \parallel P_1, \dots, \mathbf{S}_n(K_r, H_G(G_1)) \parallel F_n^1 \parallel P_n.$$

An instance of an authenticated packet stream is illustrated in Figure 4.



**Figure 4.** Authenticated packet stream.

At the receiving end, verification of the packet stream is straightforward. Assuming that at least  $m$  authenticated packets are received, the receiver can successfully reconstruct  $F^1$  and  $\mathbf{s}(K_r, H_G(G_1))$  from any combination of  $m$  packets as follows:

1. Assume that segments  $F_1^1, \dots, F_m^1$  are received. Using the  $m$  pieces, it is readily seen that

$$\mathbf{A} \cdot \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{S}_1 \\ \vdots \\ \mathbf{a}_m \cdot \mathbf{S}_1 \end{bmatrix},$$

where  $\mathbf{A} = (a_{ij})_{1 \leq i, j \leq m}$  is an  $m \times m$  matrix whose  $i$ -th row is  $\mathbf{a}_i$ .

2. Because  $\mathbf{A}$  is invertible (because of the independence condition on  $\mathbf{a}_i, 1 \leq i \leq n$ ),  $\mathbf{S}_1$  can be obtained from

$$\mathbf{S}_1 = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = \mathbf{A}^{-1} \cdot \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{S}_1 \\ \vdots \\ \mathbf{a}_m \cdot \mathbf{S}_1 \end{bmatrix}.$$

3. Using the same procedure,  $\mathbf{S}_2, \dots, \mathbf{S}_{N/m}$  can be obtained, and  $F^1$  is reconstructed by concatenating these values.

4. The same technique is applied to reconstruct  $\mathbf{s}(K_r, H_G(G_1))$ .

5. Once the reconstruction is complete, all the packets in  $G_1$  can be verified using  $F^1$  and  $\mathbf{s}(K_r, H_G(G_1))$ .

In SAIDA, the trade-off between verification probability and communication overhead can be readily governed by changing the parameters  $n$  (the number of encoded pieces) and  $m$  (the minimum number of encoded pieces needed for decoding). It should be noted that the

space overhead increase (determined by  $n/m$ ) only applies to the authentication information (i.e., hashes and signatures) and not the data itself. Furthermore, the space overhead can be adjusted adaptively by choosing several sets of the  $n$  vectors  $\mathbf{a}_i = (a_{i1}, \dots, a_{im}), 1 \leq i \leq n$  (by sender and receiver) before the start of transmission. As the loss conditions of the network change, the sender and the receivers can change  $m$  (assuming the block size is unchanged) and the corresponding set of vectors to meet the network conditions. These vectors  $\mathbf{a}_i = (a_{i1}, \dots, a_{im}), 1 \leq i \leq n$  can be chosen so that the computation of  $\mathbf{A}^{-1}$  requires  $O(m^2)$  operations [12].

## 4. Authentication probability

### 4.1. Asymptotic authentication probability

In this subsection, we derive the asymptotic authentication probability of SAIDA in a bursty loss environment. We define the *authentication probability* as  $\Pr(P_i \text{ verifiable} \mid P_i \text{ is received})$  where  $P_i$  represents the  $i$ -th packet—this definition was adopted from [10]. Using this definition, the authentication probability for a stream (consisting of multiple blocks) would be calculated by

$$\frac{1}{S} \sum_{i=1}^S \frac{\text{number of verifiable packets of block } i}{\text{number of received packets of block } i},$$

where  $S$  denotes the number of blocks within the stream. Note that this is different from the verification probability defined in Subsection 2.1. Verification probability is the number of verifiable packets of the stream divided by the total number of received packets of the stream. For example, suppose that two blocks of authenticated packets were transmitted. In the first block, five packets were received but only two were verified. In the second block, four packets were received but only two were verified. The resulting authentication probability and verification probability are  $9/20$  and  $4/9$ , respectively.

The authentication probability is directly affected by the loss model used in the analysis. In [19], it was shown that the 2-state Markov chain can accurately model bursty loss patterns in certain cases, and hence we adopt this model for our analysis. Throughout the paper, we denote this loss model as the *2-state Markov Chain (2-MC)* loss model. It is defined as follows:

**2-MC loss model:** The loss process is modeled as a discrete-time Markov chain with two states—0 and 1—representing no loss and loss, respectively. It is defined by the four transition probabilities (i.e.,  $p_{00}, p_{11}, p_{01}$ , and  $p_{10}$ ). The stationary probabilities (the long-run proportion of transitions that are into a given state) are denoted as  $\mathbf{p}_0$  and  $\mathbf{p}_1 = 1 - \mathbf{p}_0$ . The expected burst-loss

length  $\mathbf{b}$ , and probability of loss  $q$  can be expressed using the four parameters of the Markov chain.

We represent this loss process as a discrete-time binary time series  $\{S_i\}_{i=1}^{\infty}$  taking values in the set  $\{0, 1\}$  (representing no loss and loss). Before deriving the authentication probability, we need the following lemmas.

To express our main result (i.e., Proposition 1) in a more convenient form, we need to represent the four transition probabilities in terms of the stationary probabilities and  $\mathbf{b}$ .

**Lemma 1.** The four transition probabilities can be expressed using  $\mathbf{b}$ ,  $\mathbf{p}_0$ , and  $\mathbf{p}_1$  as follows:

$$p_{10} = \frac{1}{\mathbf{b}}, \quad p_{01} = \frac{\mathbf{p}_1}{\mathbf{b}\mathbf{p}_0}, \quad p_{11} = 1 - \frac{1}{\mathbf{b}}, \quad p_{00} = 1 - \frac{1}{\mathbf{b}} \left( \frac{1}{\mathbf{p}_0} - 1 \right)$$

**Proof:** Let  $\mathbf{B}$  be a random variable representing the length of a consecutive string of packet losses in steady state. Then the expected burst length is

$$\begin{aligned} \mathbf{b} &= E[\mathbf{B}] \\ &= \frac{\mathbf{m}_{00} - 1}{p_{01}}, \end{aligned}$$

where  $\mathbf{m}_{00}$  represents the expected number of transitions between successive visits to state 0. This value of  $\mathbf{m}_{00}$  can be written as an infinite sum of  $k f_{00}^k$ , where  $f_{ij}^n$  represents the probability that, starting in  $i$ , the first transition into  $j$  occurs at time  $n$ :

$$\begin{aligned} \mathbf{m}_{00} &= \sum_{k=1}^{\infty} k f_{00}^k \\ &= p_{00} + p_{01} p_{10} \sum_{k=0}^{\infty} p_{11}^k (k+2) \\ &= 1 + \frac{p_{01}}{p_{10}}. \end{aligned}$$

Substituting this value for  $\mathbf{m}_{00}$  in the above expression for  $\mathbf{b}$ , we get  $\mathbf{b} = 1/p_{10}$ . It follows that  $p_{10} = 1/\mathbf{b}$ .

Now, using the relation between stationary probabilities  $\mathbf{p}_0$  and  $\mathbf{p}_1$ , we obtain

$$\begin{aligned} \mathbf{p}_0 &= \mathbf{p}_0 p_{00} + \mathbf{p}_1 p_{10} \\ &= \mathbf{p}_0 (1 - p_{01}) + \mathbf{p}_1 p_{10}. \end{aligned}$$

Substituting the value  $1/\mathbf{b}$  for  $p_{10}$  and solving for  $p_{01}$ , we obtain

$$p_{01} = \frac{\mathbf{p}_1}{\mathbf{b}\mathbf{p}_0}.$$

The remaining transition probabilities  $p_{00}$  and  $p_{11}$  can be obtained from the relation  $p_{00} = 1 - p_{01}$  and  $p_{11} = 1 - p_{10}$ , respectively.  $\downarrow$

**Lemma 2.** Let  $T$  denote the number of transitions between successive visits to state 1 in steady state. Then the following holds:

$$E[T^2] = \frac{2p_0}{p_1} \left( \frac{1}{p_{01}} \right) + \frac{1}{p_1}.$$

**Proof:** The 2-MC loss model is an irreducible, positive recurrent Markov chain, and hence visits to a given state constitute a *renewal process*. In our model, visits to state 1 is a (renewal) event, and hence a new cycle begins with each visit to state 1. By the theory of renewal reward processes, the long-run average reward per unit time is equal to the expected reward earned during a cycle divided by the expected time of a cycle. We can form a renewal reward process by imagining that a reward is given at every time instant that is equal to the number of transitions from that time onward until the next visit to state 1. Then the expected reward earned during a cycle divided by the expected time of a cycle is given by

$$\begin{aligned} \frac{E[T + (T-1) + (T-2) + \dots + 1]}{E[T]} &= \frac{E[T^2 + T]}{2E[T]} \\ &= \frac{E[T^2]}{2E[T]} + \frac{1}{2}. \end{aligned}$$

Because the long-run average reward per unit time is the same as the average number of transitions it takes to transition into state 1, it follows that

$$\frac{E[T^2]}{2E[T]} + \frac{1}{2} = p_0 m_{01} + p_1 m_{11}.$$

Solving for  $E[T^2]$  and using the fact that  $E[T] = m_{11} = 1/p_1$ , we obtain

$$E[T^2] = \frac{2p_0 m_{01} + 1}{p_1}. \quad (3)$$

By conditioning on the next state visited,  $m_{01}$  can be obtained by solving the following linear equation:

$$m_{01} = 1 + p_{00} m_{01},$$

from which we obtain

$$m_{01} = \frac{1}{p_{01}}.$$

Substituting this result into (3) gives the desired result.  $\downarrow$

**Lemma 3.** Define  $N(n)$  as the number of visits to state 1 by time  $n$ . Then the following holds for all  $k$ :

$$\Pr\left(\frac{N(n) - \mathbf{h}}{\mathbf{s}} \leq k\right) \rightarrow \frac{1}{\sqrt{2\mathbf{p}}} \int_{-\infty}^k e^{-\frac{x^2}{2}} dx \quad \text{as } n \rightarrow \infty,$$

where  $\mathbf{h} = n\mathbf{p}_1$  and  $\mathbf{s}^2 = \mathbf{p}_1 \mathbf{p}_0 n (2\mathbf{b}\mathbf{p}_0 - 1)$ .

**Proof:** Visits to state 1 constitute a renewal event, and hence  $N(n)$  is a delayed (general) renewal process. Let  $T$  denote the number of transitions between successive visits to state 1, then by [14, Theorem 3.3.5], the following holds:

$$\Pr\left(\frac{N(n) - \mathbf{h}}{\mathbf{s}} \leq k\right) \rightarrow \frac{1}{\sqrt{2\mathbf{p}}} \int_{-\infty}^k e^{-\frac{x^2}{2}} dx \quad \text{as } n \rightarrow \infty,$$

where  $\mathbf{h} = n/E[T]$  and  $\mathbf{s}^2 = n\text{Var}(T)/(E[T])^3$ .

Using the relation  $E[T] = m_{11} = 1/p_1$ , we obtain  $\mathbf{h} = n\mathbf{p}_1$ . The variance of  $T$  is given by (using Lemma 2 for  $E[T^2]$ ):

$$\begin{aligned} \text{Var}(T) &= E[T^2] - 1/p_1^2 \\ &= \frac{2p_0}{p_1} \left( \frac{1}{p_{01}} \right) + \frac{1}{p_1} - \frac{1}{p_1^2} \\ &= \frac{2p_0 p_1 + p_{01}(p_1 - 1)}{p_{01} p_1^2} \end{aligned}$$

Using the above expression for  $\text{Var}(T)$  and  $E[T] = 1/p_1$ , we obtain

$$\mathbf{s}^2 = \frac{p_1 n \{2p_0 p_1 + p_{01}(p_1 - 1)\}}{p_{01}}.$$

Substituting  $p_{01}$  with the value derived in Lemma 1, we get  $\mathbf{s}^2 = \mathbf{p}_1 \mathbf{p}_0 n (2\mathbf{b}\mathbf{p}_0 - 1)$ , and the desired result follows.  $\downarrow$

Note that because  $\mathbf{s}^2$  is nonnegative, we conclude that  $\mathbf{b} \geq 1/(2\mathbf{p}_0)$ .

Now, we state our main result—derivation of the asymptotic authentication probability for SAIDA assuming the 2-MC loss model. As before, it is assumed that a block consists of  $n$  packets, and the minimum number of segments required for decoding is  $m$ . To derive the asymptotic authentication probability, we assume that  $n \rightarrow \infty$  and  $m = n\mathbf{p}_0 - \mathbf{g}\sqrt{n}$  for some fixed constant  $\mathbf{g}$ , so that  $m$  grows more slowly than  $n$ .

**Proposition 1.** The authentication probability of the  $i$ -th packet in the block (consisting of  $n$  packets) is given by the following expression when the minimum number required for decoding is  $m = n\mathbf{p}_0 - \mathbf{g}\sqrt{n}$ :

$$\Pr(P_i \text{ is verifiable} | P_i \text{ is received}) \rightarrow \frac{1}{\sqrt{2\mathbf{p}}} \int_{-\infty}^k e^{-\frac{x^2}{2}} dx \quad \text{as } n \rightarrow \infty,$$

where  $k = \mathbf{g}/\sqrt{\mathbf{p}_1 \mathbf{p}_0 (2\mathbf{b}\mathbf{p}_0 - 1)}$ .

**Proof:** Define the renewal process  $\{N(0), N(1), \dots\}$  as follows:  $N(0) = 0$ , and  $N(n-i)$  is the number of packet losses between  $P_{i+1}$  and  $P_n$  (the last packet in the block).

We can see that  $\Pr(P_i \text{ verifiable} | P_i \text{ is received})$  is lower bounded by the probability of the number of packet losses between  $P_{i+1}$  and  $P_n$  not being greater than  $n-m-(i-1)$ . This is because having at most  $n-m-(i-1)$  losses after  $P_i$  guarantees that we can verify  $P_i$  regardless of what happened before  $P_i$ . Hence,

$$\begin{aligned} & \Pr(P_i \text{ verifiable} | P_i \text{ is received}) \\ & \geq \Pr\{N(n-i) \leq n-m-(i-1)\}. \end{aligned}$$

Note that if  $i-1 > n-m$ , then the above inequality holds trivially. Now, let

$$y = \frac{g\sqrt{n} + i(\mathbf{p}_1 - 1) + 1}{\sqrt{\mathbf{p}_1 \mathbf{p}_0 (n-i)(2b\mathbf{p}_0 - 1)}}.$$

Then,

$$\begin{aligned} & \Pr\left\{ \frac{N(n-i) - \mathbf{p}_1(n-i)}{\sqrt{\mathbf{p}_1 \mathbf{p}_0 (n-i)(2b\mathbf{p}_0 - 1)}} \leq y \right\} \quad (4) \\ & = \Pr\{N(n-i) \leq g\sqrt{n} + \mathbf{p}_1 n - i + 1\} \\ & = \Pr\{N(n-i) \leq n-m-(i-1)\}. \end{aligned}$$

Now, using a similar approach, we find the upper bound. The authentication probability is upper bounded by the probability of the number of packet losses between  $P_{i+1}$  and  $P_n$  not being greater than  $n-m$ . This is because the verification of  $P_i$  implies that at most  $n-m$  packet losses can be tolerated after  $P_i$ . Hence,

$$\Pr(P_i \text{ verifiable} | P_i \text{ is received}) \leq \Pr\{N(n-i) \leq n-m\}.$$

Note that if  $i > m$ , then the above inequality holds trivially. Now, let

$$z = \frac{g\sqrt{n} + \mathbf{p}_1 i}{\sqrt{\mathbf{p}_1 \mathbf{p}_0 (n-i)(2b\mathbf{p}_0 - 1)}}.$$

Then,

$$\begin{aligned} & \Pr\left\{ \frac{N(n-i) - \mathbf{p}_1(n-i)}{\sqrt{\mathbf{p}_1 \mathbf{p}_0 (n-i)(2b\mathbf{p}_0 - 1)}} \leq z \right\} \quad (5) \\ & = \Pr\{N(n-i) \leq g\sqrt{n} + \mathbf{p}_1 n\} \\ & = \Pr\{N(n-i) \leq n-m\}. \end{aligned}$$

Hence, the authentication probability is lower bounded by (4) and upper bounded by (5).

Now, define

$$k = \frac{g}{\sqrt{\mathbf{p}_1 \mathbf{p}_0 (2b\mathbf{p}_0 - 1)}}.$$

Comparing the values of  $y$ ,  $z$ , and  $k$  for fixed  $i$  and  $n \rightarrow \infty$ , it follows that

$$\lim_{n \rightarrow \infty} \frac{y}{z} = \lim_{n \rightarrow \infty} \frac{z}{k} = 1.$$

Therefore, the lower bound and the upper bound for the authentication probability are asymptotically the same (for fixed  $i$  and  $n \rightarrow \infty$ ), and the following holds by Lemma 3:

$$\Pr(P_i \text{ verifiable} | P_i \text{ is received}) \rightarrow F(k) \text{ as } n \rightarrow \infty,$$

where  $F$  is a normal distribution function with a mean of zero and a variance of one. |

## 4.2. Authentication probability lower bound

In [10], the authors derived a lower bound of the authentication probability (for the *piggybacking* scheme) based on the following bursty loss model, which is motivated by ideas in the theory of error-correction codes. We denote this loss model as the *Biased Coin Toss (BCT)* loss model for reasons stated below.

**BCT loss model:** Let  $q$  be a nonnegative fraction, and  $b > 1$  an integer. For all  $i$ , a burst of length  $b$  packets begins with packet  $P_i$  (i.e., loss includes  $P_i$ ) with probability  $q$ .

Note that for  $b = 1$ , this model is equivalent to the 2-MC loss model with  $p_{01} = p_{11} = q$  and  $p_{10} = p_{00} = 1 - q$  (hence  $\mathbf{p}_1 = q$ ). This has the effect of removing the dependence of  $S_{i+1}$  on  $S_i$  (for all  $i$ ), and hence, the loss (or no loss) of a packet is determined by the toss of a  $q$ -biased coin, where the coin toss associated with each packet is independent. For  $b > 1$ , this model produces bursty loss patterns, whereas for  $b = 1$ , it produces independent packet losses.

Using the BCT loss model, we derive the lower bound of  $\Pr(P_i \text{ verifiable} | P_i \text{ is received})$  for SAIDA. According to the loss model, it is obvious that the maximum number of places where a burst error can occur (and still allow the guaranteed authentication of  $P_i$ ) is given by  $z = \lfloor (n-m)/b \rfloor$ , where  $\lfloor x \rfloor$  denotes the largest integer not greater than  $x$ . Here,  $n$  and  $m$  denote the number of encoded pieces and the minimum number needed for decoding, respectively. Because the loss of a packet is determined by the flip of a  $q$ -biased coin, the probability that  $z$  or fewer coin tosses result in losses lower bounds the authentication probability. Hence, the authentication probability is bounded as follows:

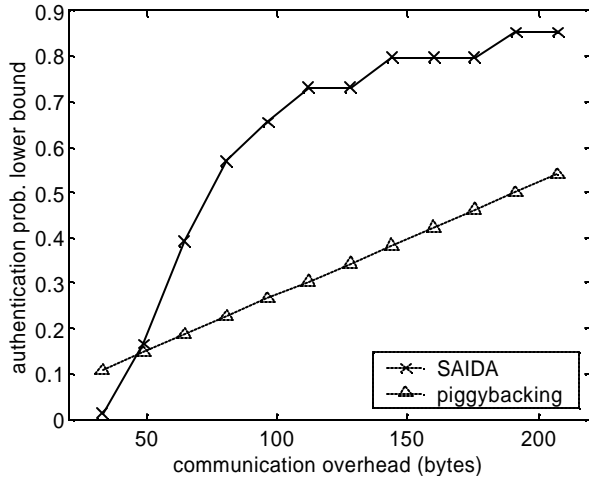
$$\begin{aligned} & \Pr(P_i \text{ verifiable} | P_i \text{ is received}) \\ & \geq \sum_{j=0}^z \binom{n-b}{j} q^j (1-q)^{n-b-j}, \text{ if } i > b-1 \end{aligned}$$

**Table 1.** Overhead comparison of the authentication schemes.

	authentication tree	EMSS	augmented chain	piggybacking	SAIDA
sender delay	$n$	1	$p$	$n$	$n$
receiver delay	1	$n$	$n$	1	$[m, n]$
computation overhead	$2n-1, 1$	$n+1, 1$	$n+1, 1$	$n+1, 1$	$n+1, 1$
communication overhead	$s + 1 + (h+1)\log_2 n$	variable	variable	variable	variable
verification probability	1.0	variable	variable	variable	variable

$$\geq \sum_{j=0}^{z'} \binom{n-i}{j} q^j (1-q)^{n-i-j}, \text{ if } i \leq b-1.$$

The above derivation takes into account the fact that none of the  $z$  coin tosses can occur in the  $b-1$  packets immediately preceding  $P_i$ , because it is assumed that the packet was received. For  $i \leq b-1$ , this would be  $i-1$  packets immediately preceding  $P_i$ .



**Figure 5.** Authentication lower bounds.

The lower bound for the piggybacking scheme is derived in a similar manner and is given below (see [10] for derivation):

$$\Pr(P_i \text{ verifiable} | P_i \text{ received}) \geq \sum_{j=0}^{z'} \binom{i-1-b}{j} q^j (1-q)^{i-1-b-j}, \text{ if } i > b+1+z'$$

$$= 1, \text{ if } i \leq b+1+z',$$

where  $z' = \min\{x_i, \lfloor b_i/b \rfloor\}$ . Parameters  $x_i$  and  $b_i$  denote the maximum number and the maximum size of the bursts that can be tolerated, respectively. Note that this lower bound was derived assuming that the signature packet was

received, whereas the lower bound for SAIDA was derived without this assumption.

In Figure 5, we plotted the lower bounds of the two schemes as the communication overhead (per packet) is increased. Because the lower bound changes with the position of the packet, the average lower bound (among  $n$  packets) was used. The following parameters were used:

- signature size = 128 bytes, hash size = 16 bytes
- $n = 128, b = 4, q = 0.2, b_i = 48$
- Parameter  $x_i$  is increased from one to twelve in increments of one.
- values for  $m$ : {67, 45, 34, 28, 23, 20, 17, 16, 14, 13, 12, 11}

## 5. Performance evaluation

### 5.1. Overhead comparison

Our main focus is minimizing the size of the overhead required to authenticate multicast packets. For our comparison, we consider only schemes that amortize a signature over multiple packets—in general, this class of schemes is more space efficient than other approaches. We compare our solution with four other previously proposed schemes—authentication tree, EMSS, augmented chain, and piggybacking scheme.

- *Sender delay*: delay on the sender side (in number of data packets) before the first packet in the block can be transmitted.
- *Receiver delay*: delay on the receiver side (in number of data packets) before verification of the first packet in the block is possible.
- *Computation overhead*: number of hashes and signatures computed by the sender per block.
- *Communication overhead (bytes)*: size of the authentication information required for each packet.
- *Verification probability*: number of verifiable packets of the entire stream divided by the total number of received packets of the stream.

Table 1 summarizes the five authentication schemes based on the performance criteria explained above. Its values were obtained based on the following assumptions:

- All five schemes employ a block size of  $n$  packets.
- Communication overhead of the authentication tree was obtained for a tree of degree two (i.e., each node of the tree has two children) and assuming a signature size of  $s$  and a hash size of  $h$ .
- The augmented chain is parameterized by the integer variables  $a$  and  $p$ , where  $p < n$ .
- For SAIDA,  $n$  is the number of encoded pieces, and  $m$  is the minimum number needed for decoding.

Note that the verification probability for the schemes EMSS, augmented chain, piggybacking, and SAIDA is not constant and actually depends on the communication overhead. The authentication tree technique has the favorable property of guaranteeing the verification of every received packet, but at the cost of a larger communication overhead—an overhead on the order of several hundred bytes would be required for practical block sizes. The authentication tree technique can also be converted to a probabilistic method by simply inserting the block signature in a subset of the packets instead of all the packets within the block. Even with this variation, the authentication tree method requires a larger communication overhead compared to SAIDA, because each packet must include several hash values needed to compute the authentication tree.

In terms of the delay involved in the authentication/verification process, it is evident that EMSS, authentication tree, and the piggybacking scheme have an advantage over the other two schemes. The authentication tree and piggybacking scheme do not require any delay for verification—each packet is verified as soon as it is received. EMSS requires no buffering on the sender side—each packet can be transmitted as soon as it is ready without the need to buffer other packets. The augmented chain technique requires the sender to buffer  $p$  packets, which is less than that required by SAIDA, the authentication tree scheme, and the piggybacking scheme. The receiver delay for SAIDA is not fixed—it could be anywhere in the interval  $[m, n]$ . It should be noted that SAIDA’s advantage over the other schemes is the ability to obtain high verification probabilities with minimal communication overhead (see Subsection 5.2). By strategy, it trades off increased delay for increased verification probability.

There is no significant difference in the computation overhead required for the four schemes. All five schemes compute only one signature per block of packets. The authentication tree technique puts a slightly heavier computation burden on the sender by requiring the computation of the authentication tree for each block of packets.

## 5.2. Verification probability vs. space overhead

As mentioned earlier, if the requirement on individual packet verification is relaxed, then the communication overhead can be reduced substantially. In this type of an approach, verification of each packet is not guaranteed and instead is assured with a certain probability. EMSS, augmented chain, piggybacking, and SAIDA fall into this category, and as expected, there is a trade-off between verification probability and communication overhead for these schemes.

For the augmented chain method, the number of hash chains per packet is not a variable parameter. However, multiple copies of the signed packet can be transmitted (for each block) to increase the probability of verification—copies would be sent with delayed intervals, because packet loss is correlated. Obviously, the size of the authentication information would increase in accordance with the number of copies sent per block. In SAIDA, higher verification probabilities can be achieved by increasing the amount of redundancy added to the data (i.e., adjusting the parameters  $m$  and  $n$ ). The tradeoff between performance and communication overhead in EMSS was already discussed in Subsection 3.1.

In Figure 6, we show the verification probability (i.e., the fraction of verifiable packets) for three probabilistic authentication schemes—augmented chain, EMSS, and SAIDA. To simulate a bursty loss environment, we used the 2-MC loss model defined in Subsection 4.1 with a packet loss probability of 0.2. The choice of 0.2 as the loss probability was motivated by the fact that, in general, the receiver loss rate is greater for multicast compared to unicast. In [19], the authors, using actual network measurements, showed that the loss rates for multicast sessions are much higher (more than twice) compared to their corresponding unicast sessions. Some multicast sessions were observed to have loss rates exceeding 20% [18, 19]. The following simulation parameters were used:

General parameters

- Packet loss probability: 0.2
- Average length of burst loss: 8 packets
- Block size: 128 packets
- Length of hash: 16 bytes
- Length of signature: 128 bytes

Parameters for EMSS

- Length of hash chain: uniformly distributed over the interval  $[1, 127]$

Parameters for the augmented chain

- $p = 6, a = 15$

Parameters for SAIDA

- values for  $m$ : {90, 80, 60, 42, 32, 26}

The solid and dashed curves represent the verification probabilities of SAIDA and the augmented chain, respectively. For EMSS, verification probabilities were obtained by simulating numerous combinations of the

three factors discussed in Subsection 3.1. The two clusters of markers represent the simulation results for EMSS—the left cluster represents EMSS implemented with two hashes appended per data packet and the right cluster represents EMSS implemented with four. Each cluster is composed of three types of markers—circle markers represent implementations with a single signature packet per block, while the triangle and asterisk markers represent implementations with two and three signature packets per block, respectively. Each type of marker was used several times to represent the different number of hashes appended in the signature packet. The number of hashes appended in the signature packet was varied from fifteen to 90 in increments of fifteen.

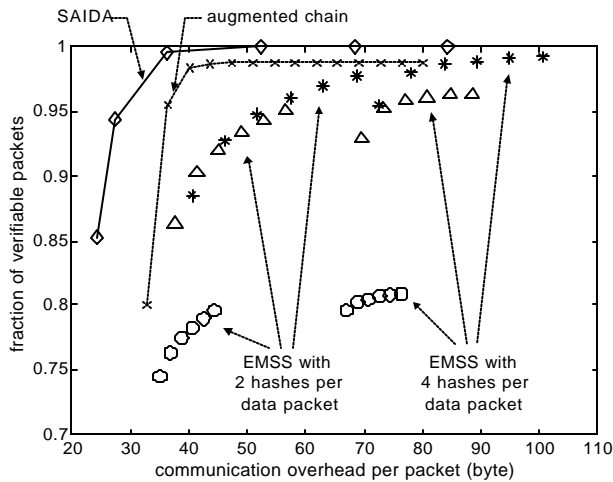


Figure 6. Verification probability.

It is apparent from the figure that SAIDA can achieve higher verification probabilities with less communication overhead, compared to the other two schemes. Unlike the schemes shown in Figure 6, the authentication tree technique can guarantee the verification of every received packet, but at the cost of a much larger communication overhead. A tree of degree two would require 248 bytes of communication overhead if the block size, hash size, and signature size are set to the same values used in the simulations of Figure 6.

When multicast packets (via UDP) are sent across networks with heavy congestion or route failures, packet loss can be high. Furthermore, conditions for the network can change abruptly during a relatively short time period. Even if the verification probability for the packets was satisfactory at the start of reception, it could deteriorate rapidly as the loss rate increases in the network. We performed experiments to examine the effect of increased packet loss on the robustness of the authentication scheme. Figure 7 shows the change in verification probability as the communication overhead is kept constant (communication overhead for the three schemes

is the same), and the packet loss probability is increased. The authentication overhead per packet was fixed at 34 bytes. Again, the 2-MC loss model is used to simulate bursty loss patterns. The following parameters were used:

General parameters

- Block size: 128 packets
- Average length of burst loss: 8 packets
- Length of hash output: 16 bytes
- Length of signature: 128 bytes

Parameters for EMSS

- Length of hash chain: uniformly distributed over the interval [1, 64]
- Number of hashes in a signature packet: 5
- Number of hashes per data packet: 2
- Number of signature packets per block: 1

Parameters for augmented chain

- Number of signature packets per block: 1
- $p = 6, a = 15$

Parameters for SAIDA

- $n = 128, m = 65$

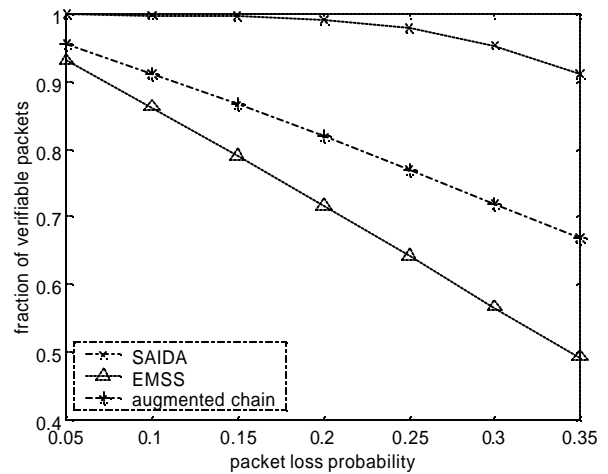


Figure 7. Verification probability with increasing loss.

In the figure, curves for EMSS and the augmented chain drop steeply to unacceptable levels when the loss rate is increased. In contrast, the curve for SAIDA drops much more moderately, maintaining a verification probability of over 0.91 when the packet loss rate is 0.35.

## 6. Conclusions

Through our results, we showed that SAIDA is an efficient method of authentication that is highly robust against packet loss. For the same amount of communication overhead, it achieved the highest verification probability among all the probabilistic authentication schemes that were examined.

Table 1 suggests that there is no single scheme that is superior in all aspects. Depending on the delay,

computation, and communication-overhead requirements, different schemes are appropriate for different applications. We expect that our scheme's high tolerance for packet loss and low communication overhead requirement will make it useful in many multicast applications—especially applications in bandwidth-limited environments (e.g., authenticated audio/video broadcasts in congested networks with high loss rates).

As already mentioned, SAIDA might not be appropriate in situations where the data to be sent is generated in real time, and immediate broadcast of it is crucial. Our scheme will be most useful in cases where the sender has *a priori* knowledge of at least a portion of the data to be broadcast (e.g., broadcast of prerecorded material).

## References

- [1] D. Boneh, G. Durfee, and M. Franklin, "Lower bounds for multicast message authentication," *Eurocrypt 2001*, May 2001, pp. 437–452.
- [2] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: a taxonomy and some efficient constructions," *IEEE INFOCOM '99*, Mar. 1999, pp. 708–716.
- [3] Y. Desmedt, Y. Frankel, and M. Yung, "Multi-receiver/multi-sender network security: efficient authenticated multicast/feedback," *IEEE INFOCOM '92*, May 1992, pp. 2045–2054.
- [4] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6, Dec. 1997, pp. 784–803.
- [5] R. Gennaro and P. Rohatgi, "How to sign digital streams," *Advances in Cryptology (CRYPTO '97)*, Aug. 1997, pp. 180–197.
- [6] P. Golle and N. Modadugu, "Authenticating streamed data in the presence of random packet loss," *Network and Distributed System Security Symposium (NDSS '01)*, Feb. 2001, pp. 13–22.
- [7] A. Koifman and S. Zabele, "RAMP: A reliable adaptive multicast protocol," *IEEE INFOCOM '96*, Mar. 1996, pp. 1442–1451.
- [8] M. Luby, M. Mitzenmacher, M. Shokrollahi, D. Spielman, and Stemann, "Practical loss-resilient codes," *ACM Symposium on Theory of Computing*, May 1997, pp. 150–159.
- [9] R. Merkle, "A certified digital signature," *Advances in Cryptology (CRYPTO '89)*, Aug. 1989, pp. 218–238.
- [10] S. Miner and J. Staddon, "Graph-based authentication of digital streams," *IEEE Symposium on Security and Privacy*, May 2001, pp. 232–246.
- [11] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," *IEEE Symposium on Security and Privacy*, May 2000, pp. 56–73.
- [12] M. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *Journal of the ACM*, Vol. 36, No. 2, Apr. 1989, pp. 335–348.
- [13] P. Rohatgi, "A compact and fast hybrid signature scheme for multicast packet authentication," *6th ACM Conference on Computer and Communications Security*, Nov. 1999, pp. 93–100.
- [14] S. Ross, *Stochastic Processes*, 2nd Edition, John Wiley and Sons, Inc., 1996.
- [15] G. J. Simmons, "Authentication theory / coding theory," *Advances in Cryptology (CRYPTO '84)*, Aug. 1984, pp. 411–431.
- [16] H. Weatherspoon, C. Wells, P. Eaton, B. Zaho, and J. Kubiawicz, *Silverback: a global-scale archival system*, Technical Report UCB/CSD-01-1139, Computer Science Division, University of California, Berkeley, Mar. 2001, 15 pp.
- [17] C. Wong and S. Lam, *Digital Signatures for Flows and Multicasts*, Technical Report TR-98-15, Department of Computer Sciences, University of Texas at Austin, May 1998, 25 pp.
- [18] M. Yajnik, J. Kurose, and D. Towsley, "Packet loss correlation in the Mbone multicast network," *IEEE Global Internet Conference*, Nov. 1996.
- [19] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modeling of the temporal dependence in packet loss," *IEEE INFOCOM '99*, Mar. 1999, pp. 345–352.