Graph-Based Authentication of Digital Streams

Sara Miner*
Dept. of Computer Science & Engineering
University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093, USA
sminer@cs.ucsd.edu

Jessica Staddon†
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
jstaddon@parc.xerox.com

Abstract

We consider the authentication of digital streams over a lossy network. The overall approach taken is graph-based, as this yields simple methods for controlling overhead, delay, and the ability to authenticate, while serving to unify many previously known hash- and MAC-based techniques. The loss pattern of the network is defined probabilistically, allowing both bursty and random packet loss to be modeled. Our authentication schemes are customizable by the sender of the stream; that is, within reasonable constraints on the input parameters, we provide schemes that achieve the desired authentication probability while meeting the input upper bound on the overhead per packet. In addition, we demonstrate that some of the shortcomings of previously known schemes correspond to easily identifiable properties of a graph, and hence, may be more easily avoided by taking a graph-based approach to designing authentication schemes.

1 Introduction

We consider the authentication of digital streams sent over a lossy network. Our model includes a single sender and a set of entities who are the intended recipients of data streams. For example, the recipients may be a multicast group receiving a feed from a central news agency, or a set of subscribers to a pay-per-view television service.

Although ensuring the privacy of the data sent in the stream is an important problem for which many useful solutions have been found (see, for example, [6, 1]), it is not the focus of our work. Instead, we seek to provide mechanisms for recipients to authenticate the data received. Specifically, they should be able to authenticate both the content itself and the source of the content. To make this possible, the sender introduces additional information into the data stream. We refer to this extra information as "authentication information".

A first attempt at solving the authentication problem involves one secret shared among all entities. The sender could use the secret as a key for a message authentication code (MAC), then MAC the content in each packet in the stream, and append the MAC to the corresponding packet. This would allow the sender to generate the authentication information quickly, and the recipients to verify the integrity of the content quickly. However, using this method, there is no source authentication. Since all parties share the same secret, any entity in the group can generate a stream that passes the authentication procedure. Furthermore, our model accommodates a dynamic recipient set in which users join and leave the group frequently. This single-secret solution would require re-keying of the entire group after each drop in membership, which is unacceptable.

A different method which provides source authentication and avoids frequent re-keying involves the use of a public-key signature scheme. The sender registers a public signing key with a certificate authority, signs the content of each packet with the corresponding secret key, and appends the signature to the packet. For each packet, recipients authenticate both packet content and source by executing the signature verification algorithm with the sender's public key. This solution, however, is costly with respect to both time (for signing and verifying) as well as bandwidth, since public-key signatures are long. In a streaming data application, signatures on each packet are too expensive to be practical.

A related approach [10, 23, 11, 18, 17] requires that the sender sign only one packet in the stream. This is the method we employ in this work. The rest of the packets in the stream are linked to that packet in a way that allows recipients to verify that they were sent by the signer. In this case, the time to run the public key signature and verifica-

*The majority of this work was completed while the author was a summer intern at Bell Labs Research Silicon Valley.

†The majority of this work was completed while the author was employed by Bell Labs Research Silicon Valley.

tion algorithms and the bandwidth used in the transmission of the signature are amortized over many packets. Assuming this "linking" of packets to the signature can be both generated by the sender and verified by recipients quickly, and the amount of extra information introduced into the stream is small, this method represents a reasonable solution.

We focus on a probabilistic model of packet loss within the network. The parameters of the model may be set in such a way that the resulting network tends to produce bursty loss, meaning packets are lost in contiguous blocks. According to several studies [5, 16], packet loss on the Internet is often bursty in nature. In addition, we consider a network model in which packets are lost independently at random, as adaptive congestion control techniques may modify the bursty loss pattern to be more random in nature [9]. Note that in either type of network, it is crucial to most of our schemes that the stream recipient actually receive the signature packet.[1] Otherwise, there may be no way to connect any of the received packets with the actual sender. For this reason, we assume that the signature packet is received. This may be accomplished with high probability by transmitting it multiple times, or empowering receivers to request re-transmission of the signature packet if it is not received. We emphasize however, that, in our model, such re-transmission requests are not allowed for other (non-signature) packets in the stream, as this might overwhelm the sender.

We take a graph-based approach to the problem of authenticating streams in a lossy network. Doing so makes properties such as overhead, delay, and the probability of authentication easier to measure and control. In particular, as explained in Section 2, overhead is correlated with the degree of the graph, the packet corresponding to a node may be authenticated when there is a path from it to the signature that only includes received nodes, and receiver delay is measured by how far forward in the stream such a path travels before it reaches the signature. Hence, graphs can be used to generate authentication schemes, as well as to determine the properties of schemes already in existence. With respect to the former case, we propose randomized and deterministic constructions, the designs of which are motivated by random and bursty network loss, respectively. Within each network model, we demonstrate a method for constructing authentication schemes that satisfy sender-prescribed constraints on overhead and the probability of authentication. In the case of a random loss network, this can be done in a tight sense by providing a formal analysis of our randomized construction. This result makes strides toward answering a more general form of an open problem posed in [18]. For the case of bursty packet loss, our constructions keep overhead at a reasonable level by correlating it with the number of bursts that must be tolerated, rather than the total amount of packet loss. In addition, although each packet tolerates bursts of a certain size and number occurring anywhere in the stream, our techniques leverage off of the highest burst tolerance attained, so that *every* packet can tolerate large bursts in some portion of the stream, with no additional overhead. Since the authentication probabilities are inputs to the scheme designs, it is possible to capture much of the priority structure that is induced by an encoding method such as MPEG [12].[2] This is advantageous, because any reduction in the authentication probability of a packet can lead to a reduction in overhead. The constructions in this paper are based on previously known chaining techniques involving hashes and MACs; however, we show the graph-based approach can make it easier to avoid the shortcomings of previously known schemes, such as low loss tolerance and the need for the sender and the receiver to be time synchronized.

OVERVIEW. The rest of the paper is organized as follows. We give details about the inputs to our schemes, as well as definitions and notation in Section 2. Authentication schemes built to tolerate random packet loss are given in Section 3. Schemes which work well in the face of bursty loss are described in Section 4. In Section 5, we discuss our main constructions, then give variations of these techniques in Section 6. Finally, we end with a review of related work and a conclusion in Sections 7 and 8, respectively.

2 Preliminaries

In this section, we provide the notation, definitions and underlying tools for graph-based authentication of digital streams. In Section 2.1, we motivate and describe the graph-based approach, provide the necessary definitions and justify the sender inputs. Section 2.2 defines the authentication tools.

2.1 Definitions and Notation

Let $\{P_1, ..., P_n\}$ denote a contiguous subset of packets in a data stream. In this paper, an authentication scheme is a directed graph with no loops and n nodes, each of which corresponds to a packet. We denote a directed edge starting at node i and ending at node j, by $\vec{e}(i, j)$. If $\vec{e}(i, j)$ is present in the graph, then the following relationship holds between packets P_i and P_j: if both the contents and source of P_j can be authenticated, then the receiver is capable of verifying the contents and source of P_i. The tools for creating

[1] The one exception to this statement is a construction discussed in Section 6.1. In this construction, it is only necessary to receive a threshold number of packets; no individual packet is necessary for authentication to be possible.

[2] See Section 2 for more on the priority structure induced by MPEG.

this relationship are defined in Section 2.2; essentially, we use hashes for leftward edges ($i > j$) and MACs for rightward edges ($i < j$). The final component of our authentication schemes is a digital signature; one of the packets, denoted by P_{sig}, is signed with a public key signature algorithm such as RSA [20]. Hence, packet P_i can be authenticated if and only if there is a path from P_i to the signature packet that only includes nodes corresponding to received packets. We denote the probability that P_i is linked to P_{sig} via such a path by $\Pr[P_i \to P_{sig}]$.

We measure the efficiency of an authentication scheme by the following parameters: overhead, delay and loss tolerance. To see how the parameters of an authentication scheme may be easily measured when it is viewed in graph form, consider the authentication scheme pictured in Figure 1.[3] Since each node has in-degree 1, the amount of overhead per packet is 1. There is no receiver delay in this scheme, as a packet may be authenticated as soon as it is received, but the sender delay (or sender buffering) is equal to 4, since the sender must buffer the contents of four packets before it can send the first one. Finally, the scheme has no loss tolerance, as the loss of a single packet makes it impossible to authenticate any later packets (no path could exist from a later packet to the signature packet, $P_{sig} = P_1$).

For every stream, we are interested in the value of $\Pr[P_i \to P_{sig} | P_i \text{ is received}]$ for $i \in \{1, ..., n\}$. In particular, we allow the sender to input desired values for these authentication probabilities. It is useful to allow a different authentication probability for each packet, because the packets in the stream may actually vary in priority. Consequently, packets deemed more important will be more tolerant to loss (because redundant authentication information will be included), and the less important packets will be less tolerant of loss, in order to avoid unnecessary overhead. As an example of how such a priority structure on packets can naturally arise, consider MPEG-2 [12]. Some MPEG-2 frames (B- and P-frames) cannot be displayed without their counterpart I-frames. When sending an MPEG-2 stream, the sender would classify I-frames as having higher importance, and our authentication schemes would then give those frames higher loss tolerance. We note, however, that while our techniques can be applied to an MPEG-2 encoding to capture much of its priority structure, the structure may not be captured *exactly*. For example, in order to ensure sufficiently high authentication probabilities, the authentication of a particular B-frame may depend on the receipt of an I-frame which MPEG does not need in order to display the frame in question.

Note that $\Pr[P_i \to P_{sig} | P_i \text{ is received}]$ is affected by the loss pattern of the network. The following model of network loss[4] is motivated by ideas in the theory of error correcting codes. (For background information on error correcting codes, see, for example, [22, 7].):

Definition 1 *Let q be a nonnegative fraction, and $b \geq 1$ an integer. In a (q, b)-network, for all i, a burst of length b packets begins with packet P_i (i.e. the loss includes P_i) with probability q.*

In a (q, 1)-network, packets are lost independently at random, whereas if $b > 1$ then the losses in a (q, b)-network are bursty in nature. Note that in Figure 1, the probability that packet P_4 can be authenticated (given that it was received) is, $(1-q)^2$ in a (q, 1)-network, since we assume the signature packet, P_1, is received.

For most applications, we assume the sender is capable of buffering a large amount of data; hence, the most important parameters are the overhead per packet, the receiver delay (i.e. the number of packets following P_i in the stream that must be sent, although not necessarily received, before P_i can be authenticated), and $\Pr[P_i \to P_{sig} | P_i \text{ is received}]$ for each packet P_i. As an example of the first two of these concepts, note that in Figure 1, there is an overhead of one hash per packet and no receiver delay. There is a strong intuition for the pairwise dependence of these parameters in general, and such dependencies certainly hold under the techniques of this paper. To make this more clear, we present the following observations. To increase the authentication probability, we may either add an edge between P_i and P_{sig} or increase the number of paths from P_i to P_{sig} through other means. Either approach increases overhead, and the latter solution may increase receiver delay. Conversely, if we decrease overhead by removing edges from the graph, this will tend to decrease the authentication probability and it may also decrease receiver delay. Finally, note that a decrease in receiver delay implies a reduction in the number of possible paths to the signature, which puts downward pressure on the authentication probability and is also likely to reduce the maximum overhead per packet. Given these dependencies, we choose to accept the maximum overhead and the desired authentication probabilities $\{p_i\}$ as inputs. Since, as we have argued, the maximum overhead and $\{p_i\}$ are dependent, we are restricted somewhat in the parameter values for which we can offer solutions (see Section 3.4 and Section 4.4). To manage receiver delay, we sign the first packet, P_1, in most of our constructions, although this alone does not prevent receiver delay. In Section 6.3, we discuss the issues surrounding moving the signature to the end, an action that generally reduces sender delay (i.e. buffering) and increases receiver delay. As a final comment, we note that if there is any receiver delay then it may be possible to mount a denial of service (DoS) attack

[3] The overall structure of this scheme is similar to one found in [10].

[4] The loss model studied here was introduced and suggested for use in this context by Mihir Bellare. Although it is different from previously proposed models [24], we believe it is useful as it produces patterns of bursty loss when $b > 1$, and it is relatively simple to work with.

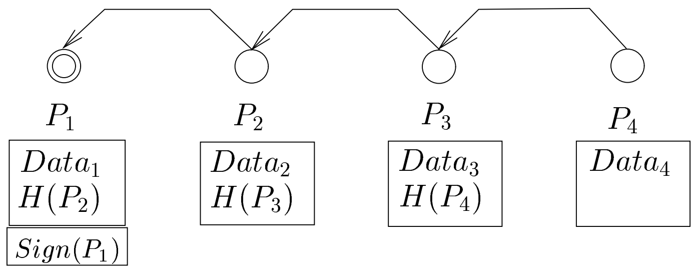

Figure 1. *A simple authentication scheme using hashes.*

on the receiver: when the authentication scheme is such that a receiver is not generally able to authenticate each packet upon receipt, then he may be forced to accept and/or store a large number of "false" packets, since he is unable to immediately determine whether or not they are valid.

2.2 Authentication Tools

THE HASH-BASED AUTHENTICATION TOOL. A public hash function (e.g., [19, 15] may be used to link the packets in a multicast stream to a signature. As discussed in Section 7, authentication schemes involving this technique appear in [10, 21, 18, 11].

Recall from Section 2.1 that if $\vec{e}(i,j)$ is present in a graph, then in the corresponding authentication scheme, the ability to authenticate P_j implies the ability to authenticate P_i. If $\vec{e}(i,j)$ is a *leftward* edge, then we can accomplish this by placing a hash of P_i in P_j. P_i may have a positive in-degree itself, indicating that hashes of other packets are included within P_i. In this case, the hash of P_i is taken after all other hashes it requires are included in it. We require strictly hash-based authentication graphs to be acyclic, so as to avoid dependencies between packets which can not be fulfilled. In addition, we note that, the strictly hash-based schemes described here have the property of non-repudiation as observed in [10, 18].

The graph in Figure 2 represents a particular hash-based authentication scheme. In the notation of Golle and Modadugu [11], it is a C_a graph with $a = 3$.

THE MAC-BASED AUTHENTICATION TOOL. Work by Perrig, et al [18, 17] makes use of message authentication codes (MACs) to link packets to a single signature packet. In the signature packet, the sender commits to a chain of keys. During each time period, one particular key is used to authenticate each packet sent out during that period. At some time later, the key is revealed in a packet further down in the stream. Hence, some time synchronization between sender and receiver is necessary, so that if a packet MAC-ed with a particular key is not received sufficiently before the key itself is revealed, that packet is thrown out and not displayed. Specifically, it is required that receivers be aware of an upper bound on the time synchronization error. This "security condition" [18, 17] must be in place, because otherwise an adversary who obtains both the MAC-ed packet and the packet containing the MAC key, can modify the data content of the former packet and then compute the correct MAC so that the modification is undetectable. In addition, we note that if the packet containing the MAC key is received before the corresponding MAC-ed packet, then the latter packet cannot be authenticated even though it has been received. In [17], the authors make this event unlikely by ensuring that the two packets in question occur far apart in the stream, at a cost of more receiver delay.

In our constructions, if $\vec{e}(i,j)$ is a *rightward* edge, meaning $j > i$, then once all the content that will make up packet P_i has been amassed (i.e. including any hash values and MAC keys) we append a MAC computed with key k_i of the contents of P_i. For ease of exposition, we will then refer to the content and the MAC as being packet P_i. We include the key k_i in packet P_j. Hence, if the source and content of packet P_j can be authenticated, then the key it contains can be used to authenticate the source and content of P_i. We suggest the use of an efficient MAC function such as HMAC [3, 2].

HYBRID SCHEMES. Finally, we make some observations on schemes in which both Hash-based and MAC-based tools are used. As described above, a graph with both leftward and rightward edges can be used to construct an authentication scheme using the leftward edges to determine where to place hashes and the rightward edges to determine where to place MACs and MAC keys. Proceeding in this way we always construct an authentication scheme that is well-defined. We note, however, that for some graphs it may be possible to reduce the use of the MAC-based tool and consequently, reduce the need for the security assumption [18, 17]. In particular, if a node i has a rightward edge, $\vec{e}(i,j)$, and is not involved in any cycles, then we can assign meaning to $\vec{e}(i,j)$ by placing a hash of P_i in P_j and still construct a well-defined authentication scheme. For a specific example of this, see Section 3.2.

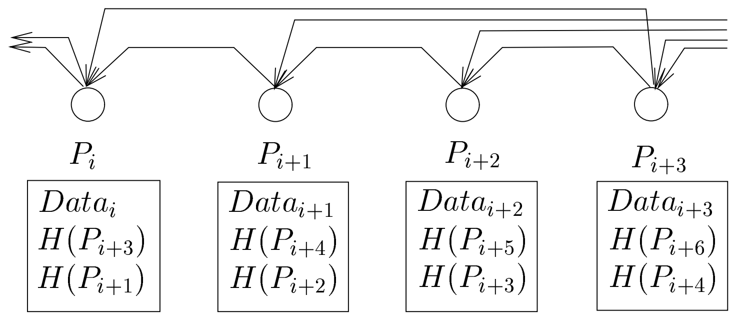

Figure 2. *A C_a graph with $a = 3$ [11]. The box below each packet represents the actual content of that packet. Although not shown, the signature packet is the first packet in the stream.*

3 Authentication Schemes Tolerant of Random Loss

In this section we consider authentication schemes in a (q, 1)-network. In such a network, each packet is lost independently at random with probability q. First, we describe an authentication scheme in which the degree of tolerance to random loss monotonically increases with packet's "distance" from the signature packet. We then prove a lower bound on the probability that any received packet, P_i, can be authenticated in a (q, 1)-network, under this scheme. This bound is tight for small values of i and goes to 1 as i goes to infinity. We note that this type of argument can easily be applied to the static hash-based scheme (EMSS) in [18]. Hence, the result represents a useful stride towards answering a more general form of the open question posed in [18].

Recall that we are most interested in applications in which the sender has a priori knowledge of the content, as is the case, for example, when the content is pre-recorded news footage. Within this scenario, there is a simple technique for modifying our basic scheme so that, for every packet P_i, we can expect P_i to be authenticated with a certain minimum probability. With this technique, we can ensure that the loss tolerance of each packet meets the sender input.

3.1 p-Random Authentication Schemes

Perhaps unsurprisingly, employing randomness when constructing authentication graphs can yield schemes without data expansion that are resistant to randomly distributed (as opposed to bursty) loss. In this section, we consider the following simple construction of a random graph and the authentication scheme it yields.

CONSTRUCTION. For a given p, $0 < p \leq 1$, we would like the n nodes in our graph to have average expected degree $p(n-1)$ (where by degree we mean in-degree plus out-degree). We number the nodes $1, 2, ..., n$. For all pairs of nodes (i, j) where $j < i$, we include a directed edge from node i to node j with probability p. We call a graph constructed in this way a *p-random graph*. We demonstrate the directed edges that may occur in a p-random graph in Figure 3.

To form the corresponding authentication scheme, we associate packet P_i with node i, and let P_1 denote the signature packet. Every edge, $i \to j$, has the following meaning: $H(P_i) \in P_j$. We call an authentication scheme constructed in this way a *p-random authentication scheme*.

Theorem 1 *With a p-random authentication scheme and no packet loss, a packet P_i, $i \geq 2$, can be authenticated with at least the probability:*

$$\Pr[P_i \to P_1 | P_i \textit{ is received}] \quad \geq \quad 1 - (1-p)(1-p^2)^{i-2}.$$

Proof We calculate the probability that node i is connected to node 1 in the corresponding p-random graph, as follows. First, with probability p, $\vec{e}(i,1)$ exists and so, node i is connected to the signature node. With probability $(1-p)p$, $\vec{e}(i,1)$ does not exist and $\vec{e}(i,i-1)$ does, so i can connect to 1 via a path from $i-1$ to 1. Proceeding in this way, we get the following expression:

$$\begin{aligned} &\Pr[P_i \to P_1 | P_i] \geq \\ &\quad p + (1-p)p \Pr[P_{i-1} \to P_1 | P_{i-1}] + \cdots \\ &\quad +(1-p)^{i-2} p \Pr[P_2 \to P_1 | P_2]. \end{aligned}$$

For small values of i it is easy to see that this expression yields the statement of the theorem. We show by induction that it holds in general. Applying the induction assumption for $1, ..., i-1$, to the right hand side of the inequality above, we have:

$$\begin{aligned} &p + (1-p)p(1-(1-p)(1-p^2)^{i-3}) + \cdots \\ &\qquad +(1-p)^{i-2}p(1-(1-p)). \end{aligned}$$

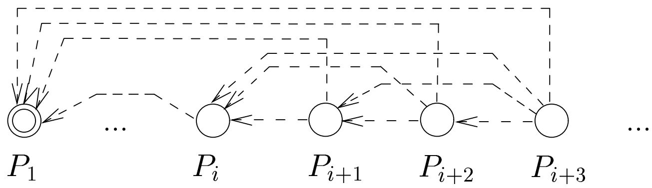

Figure 3. *In a p-random graph, each edge indicated above occurs with probability p.*

We simplify this expression by factoring out terms of the form $(1-p)$. As a first step, we have:

$$\begin{aligned}1 - (1-p)[1 - p + (1-p)p(1-p^2)^{i-3} \\ -(1-p)p + (1-p)^2 p(1-p^2)^{i-4} - \cdots \\ -(1-p)^{i-2}p + (1-p)^{i-3}p(1-p^2) \\ -(1-p)^{i-3}p + (1-p)^{i-2}p].\end{aligned}$$

Continuing to factor in this way, we eventually get:

$$\begin{aligned}&1 - (1-p)^{i-1}\left[p(1+p)^{i-3} + p(1+p)^{i-4}\right. \\ &\quad \left. + p(1+p)^{i-5} + \cdots + p(1+p) + 1 + p\right] \\ &= 1 - (1-p)^{i-1}\left[p(\frac{1-(1+p)^{i-2}}{1-(1+p)} - 1) + 1 + p\right].\end{aligned}$$

This simplifies to: $1 - (1-p)((1-p^2)^{i-2})$. □

We can use Theorem 1 to determine the authentication probabilities in a $(q,\,1)$-network. The only change to the argument above is due to the fact that packets $P_2, ..., P_n$ may be lost with probability q (since we assume that the signature packet P_1 is always received).

Corollary 1 *With a p-random authentication scheme in a $(q$, 1$)$-network, packet P_i, $i \geq 2$, can be authenticated with the probability:*

$$\begin{aligned}&\Pr[P_i \to P_1 | P_i \textit{ is received}] \geq \\ &\quad 1 - (1-p)(1-(p(1-q))^2)^{i-2}.\end{aligned}$$

Proof Because we are assuming that P_1 is always received, when we follow the same type of argument as used in the proof of Theorem 1, we get:

$$\begin{aligned}&\Pr[P_i \to P_1 | P_i] \geq \\ &\quad p + (1-p)p(1-q)\Pr[P_{i-1} \to P_1 | P_{i-1}] + ... \\ &\quad + (1-p(1-q))^{i-3}p(1-q)\Pr[P_2 \to P_1 | P_2].\end{aligned}$$

Let $\alpha_i(p) = 1 - (1-p)(1-p^2)^{i-2}$, the authentication probability found in Theorem 1. From the equality above, it follows that

$$\begin{aligned}&\Pr[P_i \to P_1 | P_i] \geq \\ &\quad \left(\frac{\alpha_i(p(1-q)) - p}{1-p}\right)(1-p(1-q)) + p(1-q).\end{aligned}$$

The statement of the theorem follows from substituting in the expression for $\alpha_i(p(1-q))$. □

3.2 Achieving a Threshold Loss Tolerance

From Theorem 1, we know that given $\epsilon > 0$, there exists i_0 such that $\forall\ i \geq i_0$, $\Pr[P_i \to P_1]$ is at least $1-\epsilon$, for some $\epsilon > 0$. Hence, we can guarantee any desired authentication probability for packets far enough away from the signature. To ensure the same minimum authentication probability for all packets, we can use the ideas of the previous section to increase the "effective distance" of the earlier nodes from the signature. The following construction describes a method for modifying the construction of the previous section so that each node in the graph has at least the effective distance of node i_0. As a result, every packet can be authenticated, given that it is received, with probability at least $1-\epsilon$.

CONSTRUCTION. Let G denote a p-random graph, on n nodes, and let $3 \leq i_0 \leq n$. For every pair of distinct nodes $(i,\,j)$, $1 < i < j \leq i_0$, we add edge $\vec{e}(i,j)$, with probability p. To form the corresponding authentication scheme, we associate packet P_i with node i, and let P_1 denote the signature packet. In such an authentication scheme the *leftward* arrows have the meaning ascribed to them by the p-random authentication scheme that forms the basis for this construction. The *rightward* edges have the following meaning: there exists a key k, such that a MAC with key k of the contents of P_i is appended to P_i, and $k \in P_j$. We call such an authentication scheme an (i_0, p)*-random authentication scheme*.

The following lemma describes how a threshold loss tolerance is achieved for all packets in an (i_0, p)-random authentication scheme.

Lemma 1 *With an (i_0, p)-random authentication scheme in a $(q, 1)$-network, each packet can be authenticated with probability at least* $1 - (1-p)[1-(p(1-q))^2]^{i_0-2}$.

Proof By Theorem 1, it suffices to show that any packet P_i, $i < i_0$, can be authenticated with the same probability as P_{i_0}. Clearly, for any path connecting P_i to the signature packet, there is a corresponding path, that exists with the same probability, connecting P_{i_0} to the signature. Hence, it remains to consider a path $\alpha = (P_{i_0}, P_{j_2}, ..., P_1)$ connecting P_{i_0} to the signature, that occurs with positive probability. If $j_2 \neq i$ then the following path has the same probability as α and connects P_i to P_1: $(P_i, P_{j_2}, ..., P_{j_r})$, $j_r = 1$. Otherwise, if $j_2 = i$, then the following path has the same probability as α and connects P_i to P_1: $(P_i, P_{i_0}, P_{j_3}, .., P_{j_{r-1}}, P_{j_r})$. $\Box$

A drawback of this scheme is that the security condition employed in [18, 17] (see description in Section 2.2) must hold. We can remove the security condition by adding the hash of each packet that is at the origin of a rightward arrow, to the signature packet. With this modification, a leftward edge from the node to the signature may be added and the initial rightward edge is unnecessary and may be removed. Of course, such modifications cause the overhead of P_i to increase. In addition, we note that unless a node is part of a cycle it is possible to use hashes instead of MACs[5], and hence, it may be possible to remove the security condition altogether, by resorting to a solely hash-based scheme. Finally, a more generally useful observation is that the security condition may only apply to a small portion of the nodes. Hence, it is possible to spend sufficient time to ensure that this small set of packets are received before sending subsequent packets at the usual higher rate. We demonstrate this last point in the following lemma.

Lemma 2 *Let $0 < r < 1$. An (i_0, p)-random authentication scheme in a $(q, 1)$-network requires the security condition ([18]) on the first $i_0 \geq \frac{\ln(\frac{1-r}{1-p})}{\ln(1-(p(1-q))^2)} + 2$ packets, to guarantee that all packets can be authenticated with probability at least r.*

Proof Recall from Theorem 1:

$$\Pr[P_i \rightarrow P_1 | P_i \text{ is received}] \geq 1 - (1-p)[1 - (1 - (p(1-q))^2)]^{i-2}.$$

Solving for i sufficiently large to ensure that this quantity is at least r yields the statement of the lemma. $\Box$

If a high authentication probability is required for all packets in a highly lossy network, and no data expansion is allowed (see Section 6.1), then the previous lemma demonstrates that the (i_0, p)-random authentication scheme is not the best authentication mechanism. However, if any of these conditions does not hold, then we believe it presents an interesting alternative to known solutions. We discuss applying the p-random authentication scheme to streams in which the desired authentication probability varies, in the following section. For discussion of other applications to which it is well suited, see Section 5.

[5] In this case only, rightward edges would indicate hashes instead of MACs.

3.3 Preserving Authentication Probabilities, $\{p_i\}$

In section 3.2, we demonstrated how to add edges to a p-random graph in order to achieve an authentication scheme with a minimum loss tolerance for all packets. As discussed in Section 2.1, there are streams for which the input authentication probabilities may cover a wide range of values. With such a stream, if we were to use the authentication methods of Section 3.2, some packets may achieve a higher authentication probability than is required. Since an increase in authentication probability implies an increase in overhead (due to the addition of edges), there is some benefit to schemes that only guarantee the required authentication probability. The overhead is minimized by first finding the minimum effective distance for each packet that gives it the required authentication probability, and then using the techniques of Section 3.2 to achieve these distances. We emphasize that because the content of one packet may depend on the content of later packets, these techniques are only useful when the sender has a priori knowledge of the content to be broadcast (e.g. pre-recorded news footage). In the following construction, we make this more precise.

CONSTRUCTION.

1. Let $\delta_{r_1} \leq \delta_{r_2} \leq \ldots \leq \delta_{r_{n-1}}$, denote the necessary effective distances for packets $P_{r_1},\ldots,P_{r_{n-1}}$ (where $P_{r_1},\ldots,P_{r_{n-1}}$ is some permutation of $P_2,\ldots,P_n$) as indicated by Lemma 1, namely, $\forall\, i$, $1 - (1-p)[1 - (1 - p(1-q))^2]^{\delta_{r_i} - 2} \geq p_{r_i}$.

2. Construct a p-random graph on n nodes.

3. Associate packet P_{r_i} with node $i+1$ in an n node graph (node 1 is reserved for the signature packet). If $\delta_{r_i} > i+1$, then add edges according to the techniques of Section 3.2 to increase the effective distance of node $i+1$ to be δ_{r_i}. The signature packet P_1 remains the first packet.

4. Reorder the nodes according to the order in which the corresponding packets are to be sent. This may cause some edges to become rightward edges. In the associated authentication scheme, the meaning of the edges is as in the construction of Section 3.2. That is, leftward edges indicate the hash-based authentication tool is to be used, and rightward edges indicate the MAC-based authentication tool is to be used, as described in Section 2.2.

In this construction, the security condition of [18] will apply whenever rightward edges appear in the corresponding graph (as discussed in Section 3.2).

3.4 Input Constraints

As discussed in Section 2, the content distributor (sender) may input the authentication probabilities $\{p_i\}$, and the overhead per packet. Using the techniques of Section 3, there is a correlation between these two parameters. Increasing the authentication probability of a packet, increases the number of edges, and hence, increases overhead. We require the sender to accept an overhead of pn, as this is the maximum expected overhead when a p-random authentication scheme is used to authenticate a stream of length n, where n is determined by $\{p_i\}$, p, and q as indicated by Theorem 1. More precisely, in order to construct a p-random authentication scheme in a (q, 1)-network with input authentication probabilities $\{p_i\}$, it is necessary that the maximum allowed overhead be at least $\left(\frac{\ln(\frac{1-p_{max}}{1-p})}{\ln(1-(p(1-q))^2)}+2\right)p$, where $p_{max} = \max_i\{p_i\}$. In such a scheme, packet P_i with input authentication probability p_i, will experience an authentication delay of $\max\left\{\frac{\ln(\frac{1-p_i}{1-p})}{\ln(1-(p(1-q))^2)}+2-i, 0\right\}$ packets.

4 Authentication Schemes Tolerant of Bursty Loss

In this section we present techniques for constructing deterministic, graph-based authentication schemes. The motivation behind the design of these schemes is resistance to the loss of contiguous blocks (i.e. bursts) of packets. Focusing on bursty loss can allow for significantly lower overhead, as the overhead in these schemes grows with the number of distinct bursts tolerated, rather than the number of packets. We build on the techniques of [11] to construct schemes that preserve a prioritization on the packets. The result is that the more important a packet is, the greater its burst tolerance, in terms of burst length and quantity. All of the packets in the stream leverage off of the burst tolerance of the most important packets, so that any packet has the same high burst tolerance as the most important packets for some portion of the stream, while being guaranteed a lower degree of burst tolerance throughout the stream. We first present constructions that are resistant to one burst only, and then describe how to modify these schemes so that they are tolerant of multiple bursts. Our descriptions are given for a regular priority structure for ease of exposition, however, the techniques may be applied to a stream with an irregular priority structure, as well. In order to determine how to construct a scheme that is consistent with the sender's authentication probabilities $\{p_i\}$, we analyze the performance of these schemes in a (q, b)-network ($b > 1$).

4.1 Piggybacking

As discussed in Section 7, Golle and Modadugu [11] has analyzed authentication schemes that are resistant to a single burst of lost packets. An upper bound B on the burst tolerance of a stream with sender packet buffer size and sender hash buffer size is given. The *burst tolerance of a stream* is the size of the maximum burst of loss such that every packet which is actually received can still be authenticated. In addition, a scheme is given with almost-optimal tolerance to a single burst.

In contrast, we consider the *burst tolerance of each packet* P_i in the stream. Here, we present schemes which achieve "better-than-optimal" burst tolerance (with respect to the bound given in [11]) for some packets, at the expense of "worse-than-optimal" burst tolerance for others. We begin by splitting the packets into priority classes, based on the values of the p_i's specified by the sender. The first node in the stream and those nodes for which the sender requests highest tolerance are placed into S_0 (the highest priority class), those with the next highest level of requested tolerance go into S_1 (the next highest class), and so on. To simplify the exposition here, we require that each of r priority classes be assigned the same number of nodes, and, furthermore, that the nodes in the highest priority class be spaced regularly throughout the stream.[6]

The intuition behind our piggybacking schemes is that we can structure the graph in such a way that the nodes in S_0 tolerate the greatest loss and do not require receipt of any nodes from lower priority classes. Then, we add edges to the graph which originate at lower priority nodes and always terminate at (or "piggyback" onto) a node in S_0. This means the hashes of lower priority packets are placed only into the nodes of the highest priority class. Note that no edges terminate at nodes which are not in the highest priority class.

CONSTRUCTION. We start with a total of n nodes representing n packets $P_1, P_2, ..., P_n$, ordered from left to right in the order the packets occur in the stream. These packets are then partitioned into r equal-sized priority classes $S_0, S_1, ..., S_{r-1}$. The size of each class is $z = n/r$. We denote the particular nodes in a class with a lower-case s whose first subscript denotes the class number, and a second subscript denoting the node's index within that class, so that the nodes in class S_0 are denoted $s_{0,1}, s_{0,2}, ..., s_{0,z}$. We assume the nodes in this highest priority class are evenly spaced throughout the stream, so that, for every $j > 0$, nodes $s_{0,j}$ and $s_{0,j+1}$ are located exactly r nodes apart in the stream. No restrictions are placed on the spacing of

[6] In general, the structure of the stream and the size of priority classes can be much more flexible, but it is always useful if (although not not required that) the highest priority nodes are spaced out in the stream. This property seems quite reasonable in streams such as MPEG.

nodes in other priority classes.

Each priority class S_i has associated with it a parameter b_i, indicating the maximum size of a burst that nodes in the class should tolerate. This value is determined based on the input of the sender, subject to constraints described in Sections 4.3 and 4.4. For a sequence of burst tolerances $b_0, b_1, ..., b_{r-1}$ we require that $\forall i$, $\exists k_i$ such that $b_i = k_i \cdot r$ and $b_0 = \max\{b_i\}$. We now explicitly state the edges which exist in the graph, and then discuss the construction below.

1. Add edges originating at the high priority nodes as follows:
   (a) [*First node in stream*] For $i = 0$ and $j = 1$, there are no edges originating at $s_{i,j}$.
   (b) [*Nodes near beginning*] For $i = 0$ and $j = 2, 3, .., k_0 + 1$, add edge $\vec{e}(s_{0,j}, s_{0,1})$.
   (c) [*General case*] For $i = 0$ and $j = k_0 + 2, k_0 + 3, ..., z$, add edges $\vec{e}(s_{i,j}, s_{i,j-1})$ and $\vec{e}(s_{i,j}, s_{i,j-1-k_0})$.
2. Add edges originating at remaining nodes as follows:
   (a) [*Nodes near beginning*] For $i = 1, 2, ..., r - 1$ and $j = 1, 2, ..., k_i$, add edge $\vec{e}(s_{i,j}, s_{0,j})$.
   (b) [*General case*] For $i = 1, 2, ..., r - 1$ and $j = k_i + 1, k_i + 2, ..., z$, add edges $\vec{e}(s_{i,j}, s_{0,j})$ and $\vec{e}(s_{i,j}, s_{0,j-k_i})$.

In Step 1 above, we begin the construction by specifying edges which originate at nodes in S_0. For the nodes in this highest priority set, the desired tolerance is $k_0 \cdot r$. The node designated $s_{0,1}$ is the first one in the stream, and it is also the signature packet. Since we assume that the signature packet is always received (see Section 1), we know that this particular packet has authentication probability 1. In general, however, for node $s_{0,j}$, $(j \neq 1)$ to tolerate any one burst of this size, two edges must originate in the node, and terminate at least $k_o \cdot r$ nodes apart. Accordingly, in Step 1(c), we insert edges $\vec{e}(s_{0,j}, s_{0,j-1})$ and $\vec{e}(s_{0,j}, s_{0,j-1-k_0})$. Note that, for nodes $s_{0,j}$, where $j = 2, 3, ..., k_0 + 1$, which are high priority nodes close to the signature node, node $s_{0,j-1-k_0}$ is not defined. In this case, according to Step 1(b), we only place the hash of each of these nodes directly into the signature packet. Each of these nodes, if received, can be authenticated with probability 1, since we assume that the signature packet itself is always received.

For each of the remaining priority classes S_i, the desired tolerance is, for some k_i, $k_i \cdot r$. Using the same ideas as above, in order to tolerate one burst of that length, we need two edges to originate at any node $s_{i,j}$, $(i \neq 0)$, and to end at nodes which are $k_i \cdot r$ nodes apart. Our edges will always terminate at nodes in the set S_0. Specifically, as described in Step 2(b), for node $s_{i,j}$, we insert edges which terminate at the first high-priority node immediately to the left of node $s_{i,j}$ (namely, $s_{0,j}$) and the node exactly $k_i \cdot r$ nodes left of $s_{0,j}$.) As above, for nodes close to the signature packet, we add edges directly to the signature instead (Step 2(a)). Specifically, for nodes $s_{i,j}$, where $j - k_i < 1$, only the edge $\vec{e}(s_{i,j}, s_{0,1})$, exists.

To form the corresponding authentication scheme from this graph, we let P_1, which is also $s_{0,1}$, be the signature node. Each edge $\vec{e}(i, j)$ in the graph implies $H(P_i) \in P_j$, just as in Section 3.1.

A simple example of our scheme with only two different priority classes (i.e., $r = 2$) is shown in Figure 4. The key observation about our scheme is that the endpoints of two edges which originate at node $s_{i,j}$ are exactly $k_i \cdot r$ packets apart in the stream, so if a burst knocks out up to b_i consecutive packets (not including packet $s_{i,j}$ itself, of course), it cannot knock out both edges. Therefore, a path is guaranteed to exist from $s_{i,j}$ to some $s_{0,j}$, and some path is guaranteed to exist from any packet $s_{0,j}$ to the signature (even for any burst of length $b_0 > b_i$). Therefore, packets in each of the priority classes are guaranteed to withstand one burst of the desired length.[7]

In effect, the lower priority nodes "piggyback" on the robust structure of edges built among the nodes in S_0. This means that if a burst that causes only nodes from classes other from S_0 to be lost, the receiver's ability to authenticate any packets that were not lost in that burst is not affected. It is most damaging to the stream when nodes from the high priority class are lost, since their loss may hinder the receiver's ability to authenticate additional nodes. However, nodes in S_0 are spaced throughout the stream specifically to minimize the number of them lost in any one burst, and, recall that the graph was built so that even in the face of considerable loss, the receiver is still likely to be able to authenticate them. Furthermore, because we have classified the non-S_0 nodes as lower priority, if the loss of several S_0 nodes causes the sender to be unable to authenticate some of them, presumably it is not particularly detrimental to the receiver's use of the data in the rest of the stream.

4.2 Tolerating Multiple Bursts

The construction presented above tolerates single bursts. We now extend the scheme to tolerate multiple bursts. In order to tolerate x_i bursts of size at most b_i, packet P_i in the above scheme requires out-degree at least $x_i + 1$. The length of bursts the packet must tolerate determines the distance between the endpoints of its $x_i + 1$ out-edges. Note that, in the original scheme above, the endpoints of the out-edges are a distance $k_i \cdot r$ apart to achieve tolerance of a single

[7] In fact, since all of the edges in the graph are directed toward the signature packet, a particular packet in our stream can tolerate unrestricted bursts after it in the stream.

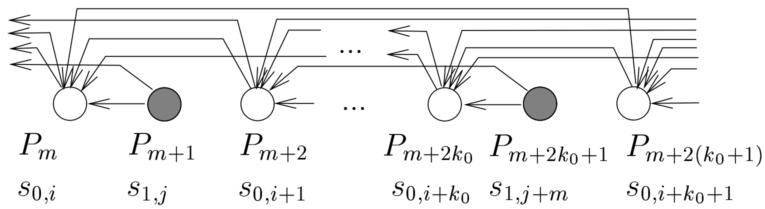

Figure 4. *A piggybacking scheme with $r = 2$. The highest priority class of nodes is white, and nodes in the other class are shaded. The signature is on the first packet in the stream.*

burst as long as $k_i \cdot r$ packets. To tolerate multiple bursts, then, we insert the edges described below. This construction allows packet $P_{i,j}$, when received, to tolerate up to x_i bursts of as many as $k_i \cdot r = b_i$ lost packets.

1. Add edges originating at the high priority nodes as follows:
   (a) [*First node in stream*] For $i = 0$ and $j = 1$, there are no edges originating at $s_{i,j}$.
   (b) [*Nodes near beginning*] For $i = 0$ and $j = 2, 3, .., x_0 k_0 + 1$, add edge $\vec{e}(s_{0,j}, s_{0,1})$.
   (c) [*General case*] For $i = 0$ and $j = x_0k_0 + 2, x_0k_0 + 3, ..., z$, add edges $\vec{e}(s_{i,j}, s_{i,j-1})$, $\vec{e}(s_{i,j}, s_{i,j-1-k_0})$,..., $\vec{e}(s_{i,j}, s_{i,j-1-k_0 x_0})$.
2. Add edges originating at remaining nodes as follows:
   (a) [*Nodes near beginning*] For $i = 1, 2, ..., r - 1$ and $j = 1, 2, ..., x_i k_i$, add edge $\vec{e}(s_{i,j}, s_{0,j})$.
   (b) [*General case*] For $i = 1, 2, ..., r - 1$ and $j = x_i k_i + 1, x_i k_i + 2, ..., z$, add edges $\vec{e}(s_{i,j}, s_{0,j})$, $\vec{e}(s_{i,j}, s_{0,j-k_i})$,..., $\vec{e}(s_{i,j}, s_{0,j-x_i k_i})$.

4.3 Preserving Authentication Probabilities, $\{p_i\}$

In a scheme constructed as in Section 4.2, there are two parameters associated with each packet P_i: the number of bursts, x_i, that P_i can tolerate, and the size of each burst, b_i. We need to determine the values of x_i and b_i such that $\Pr[P_i$ can be authenticated $|P_i$ is received$] \geq p_i$. If P_i is received but cannot be authenticated, then it must be that, within the nodes between the signature and P_i itself, either a burst of length greater than b_i has occurred, or more than x_i bursts of size at most b_i have occurred. If neither of these events occurs, then the largest number of places where the (q, b)-network loss model could have indicated that a block of lost nodes of length b should start can be expressed by $y = \min\{x_i, \lfloor \frac{b_i}{b} \rfloor\}$. [8]

Therefore, the probability that y or fewer coin flips result in losses, lower bounds the expression we want when $i - 1 - b > y$, and when $i - 1 - b \leq y$, P_i must share an edge with the signature packet, so we know its authentication probability is 1. We take into account the fact that none of these y flips can happen in the $b - 1$ nodes immediately preceding P_i itself, since we know that P_i was received. Therefore, the probability that P_i can be authenticated, given that it is received, is

$$\begin{aligned}
&\geq \sum_{s=0}^{y} \left[\binom{i-1-b}{s} q^s (1-q)^{i-1-b-s}\right] \\
&= (1-q)^{i-1-b-y} \left[\binom{i-1-b}{0} q^0 (1-q)^y \right.\\
&\quad + \binom{i-1-b}{1} q^1 (1-q)^{y-1} + ... \\
&\quad \left. + \binom{i-1-b}{y} q^y (1-q)^0\right] \\
&\geq (1-q)^{i-1-b-y} \left[\binom{y}{0} q^0 (1-q)^y \right.\\
&\quad \left. + \binom{y}{1} q^1 (1-q)^{y-1} + ... + \binom{y}{y} q^y (1-q)^0\right] \\
&\geq (1-q)^{i-1-b-y}
\end{aligned}$$

Solving $(1-q)^{i-1-b-y} \geq p_i$ yields

$$y \geq i - 1 - b - \frac{\ln(p_i)}{\ln(1-q)}$$

This indicates that, given appropriate overhead allowances, the sender's inputs can be satisfied if either $x_i \geq i - 1 - b - \frac{\ln(p_i)}{\ln(1-q)}$ and $b_i \geq b\left(i - 1 - b - \frac{\ln(p_i)}{\ln(1-q)}\right)$ hold. We note that these bounds function as a guide only, as for some values of the parameters, the proven bounds are sufficient but not necessary.

[8] Here, we do not necessarily mean that y distinct bursts of length b occur in the stream; instead, we are counting each node where, in our loss model, the flip of a q-biased coin indicated that the next block of b nodes was lost. This is independent of coin flips associated with any other nodes in the stream.

4.4 Input Constraints

We additionally constrain the sender's input so that the overhead requirement is consistent with the required parameters x_i and b_i as determined in Section 4.3. In our piggybacking scheme tolerating multiple bursts (Section 4.2), overhead contributed by each node is directly correlated with x_i, the number of bursts that each node must tolerate. The number of edges in the graph is at most $\sum_{i=2}^{n}(x_i+1)$. Each of these edges represents a hash placed in one of the z nodes in the highest priority set. So the high priority nodes must tolerate $\frac{1}{z}\sum_{i=2}^{n}(x_i+1)$ hashes on average.[9] The other nodes in the stream incur no overhead. Note that increasing the sizes of the burst that nodes tolerate does not increase the total overhead in the system. The longer the bursts tolerated, though, the more nodes which must have hashes placed in the signature packet itself.

5 Discussion

In this section we consider the merits of the two classes of authentication schemes studied in this paper. Variations on these basic methods are discussed in Section 6.

ON THE RANDOMIZED APPROACH. As mentioned earlier, the randomized construction is most efficient when either it is unnecessary for all authentication probabilities to be high, or the network over which the data is streamed is not highly lossy (i.e. q is relatively small). The first condition may naturally arise due to the type of data being streamed (e.g. MPEG files), also it may be achieved by adding redundancy to the stream prior to the authentication structure (see Section 6.1), as this allows the authentication probabilities for each packet to be reduced. The goals of our randomized schemes are most similar to those of the hash-based construction (EMSS) of [18]. However, it is difficult to compare the performance of our constructions with those of [18] because, although simulation data was collected for the latter, no formal analysis was completed. Also, our p-random authentication scheme can be viewed as a generalization of EMSS, as in EMSS, the number of packets in which the hash of a packet may be included is fixed, with the actual choice of the "carrier" packets (i.e. endpoints of edges) being made either deterministically or randomly.

If it is decided to augment the authentication probabilities in the p-random authentication scheme and create a (i_0, p)-random authentication scheme, then we can no longer claim non-repudiation in general and, consequently, the authentication scheme becomes somewhat similar to the MAC-based scheme, TESLA, of [18, 17]. As mentioned earlier, TESLA achieves essentially unbounded loss tolerance with very low overhead at the cost of a security condition (see Section 2.2) that holds for all packets. Although, our construction certainly cannot compete with TESLA in terms of overhead and overall loss tolerance, we believe that it is an interesting alternative for some applications as a high authentication probability can be guaranteed for many of the packets in the stream, while the security condition may only be applicable to a small portion of the stream. For example, if we consider a stream of 100 packets in which the packet size is 512 bytes, and network loss is 10% ($q = .1$), we can use a (50, .2)-random authentication scheme to achieve an authentication probability of at least 80% for all the packets, while requiring a security condition on at most the first half of the packets in the stream. The authentication probability rises to more than 90% for at least 35% of the stream. The overhead we incur is expected to be no more than 200 bytes per packet ($< 50\%$), and it is expected to be significantly less for most of the packets. For example, packet P_i, $i \geq 51$ is expected to incur no more than $2(100 - i)$ bytes of overhead.

ON THE DETERMINISTIC APPROACH. In our piggybacking schemes, we leverage on the concept of prioritized packets in order to reduce the overall authentication overhead in the stream. The schemes are straightforward to implement, and, when the signature is on the first packet in the stream, they provide non-repudiation. This scheme is particularly attractive when the sender wants significant control over the tolerance that each packet has. Furthermore, even those packets which are given low priority have, for most instances of loss, tolerance as high as the highest priority packets in the stream. This is due to the fact that, in a node $s_{i,j}$'s path to the signature node, the first hop always leads to a node in the high priority set S_0. The path from any high-priority node to the signature includes only on high-priority nodes. This means that, assuming that at least one out-edge exists from $s_{i,j}$, a path from it to the signature will exist even if up to x_0 bursts of size at most b_0 occur.

Finally, we briefly mention that in the case where $b = 1$ the constructions resulting from the approach of Section 4 may be significantly less efficient than the corresponding randomized constructions, largely because the lower bounds proven in Section 4.3 are less tight. However, in a network where losses are bursty ($b >> 1$), the overhead required from our piggybacking schemes will be significantly less than the randomized schemes.

6 Variations on Our Schemes

In this section, we discuss some variations on the authentication methods of this paper. We begin by considering how redundancy can be combined with the known authentication techniques, then we consider a variation of the

[9] The overhead of the signature packet is determined slightly differently, due to the nearby nodes who each have one hash included in it. Depending on the x_i and b_i values associated with these nodes, the overhead placed into the signature packet will vary.

piggybacking method that is most useful in the case of a single burst, and a technique for shifting between receiver and sender delay.

6.1 Using Redundancy

We consider two methods for constructing authentication schemes with redundancy encoding[10] by redundantly encoding one of the previously discussed authentication schemes, and by redundantly encoding the stream itself before adding authentication information. In either case, we propose to use Tornado codes [13] for the redundancy encoding as they are efficient in terms of computation and expansion. Specifically, for a stream of n packets, Tornado codes increase the number of of packets that are transferred to $\frac{n}{1-p(1+\epsilon)}$, where p and ϵ are positive fractions, in exchange for the property that if any p fraction of the expanded set of packets are lost, then the n original packets may be reconstructed in time proportional to $n \ln(1/\epsilon)$.

As discussed below, there are advantages and disadvantages to either approach. We stress, however, that in either case, any authenticated packet is as useful as any other in terms of authenticating the stream. This is to be contrasted with all the previous (no expansion) schemes, in which the signature packet is absolutely necessary for the authentication of the stream.

AUTHENTICATION STRUCTURE FIRST, REDUNDANCY ENCODING SECOND. Here we consider the following approach to combining authentication with redundancy encoding: construct an authentication scheme as described in the previous sections, and then redundantly encode the resulting packets and transmit this expanded set of packets. When the receiver obtains a threshold number of the packets, she will be able to reconstruct the entire stream and authenticate it as before. The advantage of this approach is that we may utilize a low overhead authentication scheme because packet loss is less of an issue. For example, we may employ a simple hash-based authentication scheme as shown in Figure 1. A drawback to this scheme is that it introduces significant receiver delay, as the receiver must now receive slightly more than n packets (provided the encoding is done with Tornado codes) before a single packet can be displayed. In addition, the scheme is vulnerable to denial of service (DoS) attacks because there is no way to authenticate packets as they are received, and so, an attacker may flood the receiver with many packets before the receiver is capable of determining that the packets are bogus.

REDUNDANCY ENCODING FIRST, AUTHENTICATION STRUCTURE SECOND. An alternate approach is to apply redundancy to the stream first. Then the authentication problem is similar to before, namely, how to add authentication information to the packets in a manner that is has a tolerable level of overhead and delay. The importance of resistance to loss is somewhat lessened because of the packet expansion. As above, this approach adds to receiver delay because it is necessary to receive slightly more than n "good" packets before the original stream can be recovered and displayed. This method does yield a scheme that is resistant to DoS provided the authentication structure used does not incur significant receiver authentication delay.

[10] We point out that Perrig, et al [17] make use of redundancy in an extension of the EMSS scheme, redundantly encoding on a hash-by-hash basis.

6.2 Bidirectional Piggybacking

Consider the simple piggybacking construction described in Section 4.1, which tolerates only a single burst. When the signature is at the beginning of the stream, the two edges that originate at any packet $P_{i,j}$ which is not a member of the highest priority class both terminate at packets which come before it in the stream. In a model in which we assume no more than one burst, any packet which is not lost in the burst itself must either be before the entire burst or after the entire burst. For this reason, it is useful to modify the piggybacking idea slightly, so that one edge terminates earlier in the stream, and the other terminates later in the stream. In Figure 5, we give an example of a bidirectional piggybacking graph . In this way, at most one of the packets which contains the hash of the packet in question will be lost in a burst. If the burst occurs after the packet in question, the path to the signature will proceed directly leftward, toward the signature. On the other hand, if the burst is before the packet in question so that the immediate leftward path is knocked out, the path will be forced to move rightward first. Then the burst tolerance of the packet in question is determined by the tolerance of the one packet to its right which contains its hash. We note that our construction contains no cycles, so all edges in our graph represent hashes.

6.3 Reversing Any Authentication Graph

In any hash-based authentication scheme like those we have described, the option exists to place the signature packet at the end of the stream or at the beginning. The tradeoff involves where buffering occurs and how quickly full authentication occurs for each packet. To illustrate, we consider two extreme examples. First, in the schemes presented in this paper, the signature is computed on the first packet in the stream and all edges are directed leftward. As a result, the sender must have access to the entire content for the stream before the first packet is sent, because hashes of packets that come later in the stream must be included in the signature packet. This could be a drawback when the data to be sent is generated in real-time, and immediate dissem-

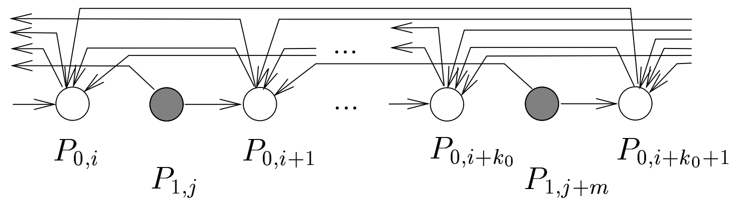

Figure 5. *Bidirectional piggybacking with $r = 2$, and the signature on the first packet in the stream.*

ination of it is crucial. On the other hand, the benefit of this approach is that receivers, upon getting a packet, can immediately authenticate the content, and confirm the sender (i.e., check the hashes that link this packet to the signature packet, which was received earlier in the stream). It is not necessary for the receiver to delay the use of any packet (except while its hash is checked against values the receiver *already* has).

In contrast, consider the other extreme, where the signature is computed on the last packet in the stream, and all edges are directed rightward. This is useful in settings where data is generated in real-time, since senders can send a packet immediately after the data becomes ready. However, receivers, upon getting a packet, will be unable to confirm the sender's identity until the signature packet, the last packet in the stream, is received. This may cause the receiver to delay his usage or display of the data, resulting in the need for a large buffer to store the entire stream until transmission is complete.

Clearly, there is a tradeoff here, and distributors of different applications may select different options. We chose to present our schemes in such a way that receiver authentication delay is minimized, which we believe to be desirable in most cases.

In addition, we sketch here how a particular scheme may be changed from one extreme to the other. Any graph which represents a valid authentication scheme with the signature packet at one end can be "reversed", so that the signature is computed on the packet at the other end of the stream. The new graph will consist of the same n nodes, but the signature packet is moved to the other end. Additionally, edges are reversed in such a way that, for every path $(P_{i_1}, ..., P_{i_r}, P_{sig})$ existing in the original graph, the following is a path in the "reversed" graph: $(P_{i_r}, ..., P_{i_1}, P_{sig})$. We remark that in graphs where both leftward and rightward edges exist, our technique will not eliminate sender buffering altogether.

7 Related Work

The problem of authenticating digital streams with computational security was initially studied by Gennaro and Rohatgi [10], who proposed a model in which the sender signs the first block of a message, then ties in subsequent blocks in a way that guarantees the property of non-repudiation. For a stream which is finite and known in advance, the sender inserts the hash of each block into the block that precedes it. In the case of an infinite stream, multiple instances of a one-time signature scheme are used. One-time signatures are advantageous because they are more efficient than regular public key schemes. The solutions in [10], however, do not tolerate packet loss, and the size of one-time signature keys and signatures, themselves (both of which contribute to overhead) is quite large.

In a later paper [21], Rohatgi extends the work of [10], by making use of k-time signature schemes, where one pair of keys can be used in k signatures instead of just one, but signing and verifying can still be performed relatively quickly. Further speedup is achieved by having the keys computed and certified off-line. The k-time keys are sent in more than one packet, addressing the reliability concerns of the earlier scheme. Other optimizations address efficiency concerns.

Within the same model, Wong and Lam [23], use ideas from [14] to form a tree-based authentication scheme. The packets of the stream form the leaf nodes and parent nodes are computed as the hashes of their children, with a signature at the root. Extra information is sent with each packet to allow authentication of the entire chain before all packets whose hashes are in the tree arrive, and caching is used to make the verification process more efficient. The Wong-Lam scheme has no receiver authentication delay, and the size of the packet to be signed is relatively small, however the scheme suffers from a high amount of overhead on the non-signature packets, as $O(\log n)$ extra pieces of information are appended, where n is the number of packets in the stream.

Based on the observation that a significant amount of the packet loss on the Internet occurs in bursts [5, 16], Golle and Modadugu [11] constructs a hash-based scheme whose goal is the authentication of received packets in the face of a single burst of loss . The scheme includes a signature on the final packet in the stream, and the hash of each of the other packets is placed in several packets in the stream. He presents bounds on the overall burst tolerance possible in this model given particular efficiency resources, and shows his scheme to be optimal in that respect.

TESLA[11], a scheme developed by Perrig, et al [18, 17] is a MAC-based stream authentication scheme which tolerates essentially unbounded, random loss. The first packet in the scheme is signed by the sender, and includes a commitment to a chain of MAC keys. Each subsequent packet P_i is MAC-ed with a key K_i, and each K_i is revealed in some packet later in the stream. A drawback of the scheme is that it requires time synchronization between the sender and each receiver, as the receiver must be assured that packet P_i is received before packet P_{i+1} is sent (see the security condition described in Section 2.2). We use techniques found in TESLA to increase the authentication probability of a p-random authentication scheme. However, we are able to reduce the number of packets over which the security condition applies while maintaining a high probability of authentication. The cost of this is a higher overhead than that incurred in TESLA.

In addition, [18] studies some interesting hash-based authentication schemes, all of which are easily represented in the graph-based framework. Our p-random construction can be viewed as a generalization of EMSS. In EMSS, it is decided a priori how many packets will contain $H(P_i)$, for each i. The actual location of these packets is then chosen either deterministically or randomly. No formal analysis of EMSS is provided in [18], however the technique used in Theorem 1 of this paper can be applied to the static form of EMSS. Hence, Theorem 1 makes strides toward answering a more general form of an open question posed in [18].

Finally, using a different model, Canetti et al [6] give a scheme suitable for authenticating multicast streams, where each packet is independently verifiable. Each receiver is given a different subset of k out of the universe of K MAC keys, and the sender sends each packet in the stream along with its MAC computed under each of the K keys. A receiver verifies the k MACs for which it has the corresponding key. If any of the checks fail, the receiver rejects the packet. Otherwise, the receiver is assured of the authenticity of the content assuming that there are no receiver coalitions of more than some bounded size. This scheme has the desirable property that each packet can be verified upon receipt, independently of any other packets, however the authentication overhead per packet may be large. If we seek to guarantee that no set of w users can cause another user to incorrectly believe a packet is authentic, the the authentication overhead roughly grows as $O(m^{w/k})$ MACs, when $w \leq k$, where m is the number of receivers and w is the maximum coalition size. This bound on the overhead follows from the fact that the scheme in [6] requires that MAC keys be allocated according to a "cover-free family" [8] to achieve such a security guarantee. We note that graph based techniques do not appear to be useful in analyzing schemes of this type, as authentication information is not shared amongst the packets. Boneh et al [4], have recently made strides in analyzing schemes of this type through alternate methods.

[11]TESLA cannot be naturally represented in the graph-based model we study here.

8 Conclusion

Our work unifies many of the ideas in [10, 21, 11, 18, 17] in that we present a framework within which many of these constructions can be represented. In addition, we allow priorities to be placed on packets, a desirable feature, as this is often a consequence of the encoding methods used for digital streams. We demonstrate that these priorities may be preserved by the construction in a tight sense when the network loss pattern is random. For bursty networks, we provide burst-tolerant authentication schemes that minimize overhead by tightly correlating it with the number of bursts that must be tolerated, rather than with the total amount of loss. In addition, these schemes take maximal advantage of the high burst tolerance that must be guaranteed for some packets, by using "piggybacking" techniques to allow *all* packets to have a high burst tolerance over some portion of the stream, with no additional overhead. These constructions for the bursty case, as well as the p-random constructions for the random loss case, incur no receiver delay, making them resistant to denial of service attacks.

There is still much to be done towards forming a comprehensive theory of graph-based authentication schemes. For example, tight lower bounds relating the scheme parameters in either the bursty or random network loss cases are unknown, to the best of our knowledge.

9 Acknowledgments

The authors would like to thank Martín Abadi, Mihir Bellare, Adrian Perrig, and the anonymous referees for helpful comments on this work. More specifically, we acknowledge Mihir Bellare for suggesting the network loss model we describe in Section 2.1.

Sara Miner was supported in part by Mihir Bellare's 1996 Packard Foundation Fellowship in Science and Engineering.

References

[1] M. Abdalla, Y. Shavitt, and A. Wool. *Key Management for Restricted Multicast Using Broadcast Encryption.* In IEEE/ACM Transactions on Networking **8** (2000), pp 443–454.

[2] M. Bellare, R. Canetti, and H. Krawczyk. *Keying Hash Functions for Message Authentication.* In Advances in

Cryptology - Crypto 96, Lecture Notes in Computer Science Vol. **1109** (1996).

[3] M. Bellare, R. Canetti, and H. Krawczyk. *HMAC: Keyed-Hashing for Message Authentication.* IETF Internet Request for Comments 2104, February 1997.

[4] D. Boneh, G. Durfee and M. Franklin. *Lower Bounds for Multicast Message Authentication*. To appear in the proceedings of Eurocrypt 2001.

[5] M. Borella, D. Swider, S. Uludag and G. Brewster. *Internet Packet Loss: Measurement and Implications for End-to-End QoS*. In Proceedings, International Conference on Parallel Processing, August 1998.

[6] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas. *Multicast Security: A Taxonomy and Some Efficient Constructions*. In IEEE Infocom, **2** (March 1999), pp 708–716.

[7] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[8] P. Erdös, P. Frankl and Z. Füredi. *Families of Finite Sets in which No Set is Covered by the Union of r Others*. Israel Journal of Mathematics **51** (1985), pp 75–89.

[9] S. Floyd and V. Paxson. *Why We Don't Know How to Simulate the Internet*. In Proceedings of the 1997 Winter Simulation Conference. Atlanta, Georgia, 1997.

[10] R. Gennaro and P. Rohatgi. *How to Sign Digital Streams*. In Advances in Cryptology - CRYPTO '97, Lecture Notes in Computer Science **1294** (1997), pp 180–197.

[11] P. Golle and N. Modadugu. *Authenticating Streamed Data in the Presence of Random Packet Loss*. ISOC Network and Distributed System Security Symposium (2001), pp 13–22.

[12] ISO/IEC International Standard 13818, Parts 1-9. *Information technology - Generic coding of moving pictures and associated audio information*.

[13] M. Luby, M. Mitzenmacher, M.A. Shokrollahi, D. Spielman and V. Stemann. *Practical Loss-Resilient Codes*. 29th Symposium on the Theory of Computing, ACM (1997), pp 150–159.

[14] R. Merkle. *A Certified Digital Signature*. In Advances in Cryptology - CRYPTO '89, Lecture Notes in Computer Science **293** (1990), pp 218–238.

[15] National Institute of Standards and Technology. *The Secure Hash Standard*. NIST FIPS Pub 180-1, 1995.

[16] V. Paxson. *End-to-End Internet Packet Dynamics*. In IEEE/ACM Transactions on Networking **7** (1999), pp 277–292.

[17] A. Perrig, R. Canetti, D. Song and J.D. Tygar. *Efficient and Secure Source Authentication for Multicast*. ISOC Network and Distributed System Security Symposium (2001), pp 35–46.

[18] 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 (2000), pp 56–73.

[19] R. Rivest. *The MD5 Message Digest Algorithm*. Internet Request for Comments, April 1992.

[20] R. Rivest, A. Shamir and L. Adleman. *A Method for Obtaining Signatures and Public-Key Cryptosystems*. Communications of the ACM, (1978), pp 120–126.

[21] P. Rohatgi. *A Compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication*. 6th ACM Conference on Computer and Communications Security (1999), pp 93–100.

[22] J. H. van Lint. *Introduction to Coding Theory*. Third Edition, Springer-Verlag, 1998.

[23] C. K. Wong and S. Lam. *Digital Signatures for Flows and Multicasts*. In IEEE/ACM Transactions on Networking **7** (1999), pp 502–513.

[24] M. Yajnik, S. Moon, J. Kurose and D. Towsley. *Measurement and Modelling of the Temporal Dependence in Packet Loss*. In IEEE Infocom '99.