# Authenticating Streamed Data in the Presence of Random Packet Loss

**(Extended Abstract)**

Philippe Golle*
pgolle@cs.stanford.edu

Nagendra Modadugu
nagendra@cs.stanford.edu

## Abstract

*We propose a new scheme for authenticating streamed data delivered in real-time over an insecure network. The difficulty of signing live streams is twofold. First, authentication must be efficient so the stream can be processed without delay. Secondly, authentication must be possible even if some packets in the sequence are missing. Streams of audio or video provide a good example. They must be processed in real-time and are commonly exchanged over UDP, with no guarantee that every packet will be delivered. Existing solutions to the problem of signing streams have been designed to resist worst-case packet loss. In practice however, network loss is not malicious but occurs in patterns of consecutive packets known as bursts. Based on this realistic model of network loss, we propose an authentication scheme for streams which achieves better performance as well as much lower communication overhead than existing solutions. We have implemented our constructions as plug-ins to the RealSystem platform from Real Networks to authenticate audio and video streams.*

**Keywords:** authentication, non-repudiation, streams.

## 1. Introduction

Video and audio documents are delivered over a network as a continuous sequence of packets (a stream.) We would like a signature scheme that allows two parties to exchange a stream with guarantees of integrity and non-repudiation. Consider a radio station broadcasting over the Internet. It is important to listeners to have guarantees that the audio stream they receive was generated by the station. It is equally important to the station that only content it generated be attributed to it. For example, malicious parties should be prevented from injecting commercials or offensive material into the stream.

There are two issues to consider when signing streams. On the one hand, the signature scheme must be efficient enough to permit authentication on the fly without introducing delays. On the other hand, the signature scheme must be robust enough that authentication remains possible even if some packets are lost.

The naive solution to authenticate a stream is to sign each packet in the stream individually. The receiver checks the signatures of packets as they arrive and stops processing the stream immediately if an invalid signature is discovered. Immediate authentication is possible, but the computational load on both the sender and the receiver is too great to make this approach practical.

A more efficient solution is proposed in [4] by Gennaro and Rohatgi. They observe that one-time signatures can be used in combination with a single digital signature to authenticate a sequence of packets. Each packet carries a public-key, which is used in a one-time signature scheme to sign the following packet. Only the first packet needs to be signed with a regular digital signature. Since one-time signatures are an order of magnitude faster to apply than digital signatures, and can also be verified somewhat more efficiently, this solution offers a significant improvement in execution speed.

However, there is a major difficulty with this approach. Recall that audio and video streams are sent using UDP, which provides only "best-effort" service and does not guarantee that all packets will be delivered. If a packet is missing, the authentication chain is broken and subsequent packets can not be authenticated. (Another problem is that one-time signatures incur a substantial communication overhead[1].)

Let us examine the issue of packet loss in more detail. If a sequence is received incomplete, we would still like to be able to authenticate all the packets that were not lost. This defines resistance to loss in a strong sense: a packet is either lost or authenticable. A weaker alternative would allow a few packets to be received unauthenticated in case of packet loss. We offer two justifications

* Supported by Stanford Graduate Fellowship

[1] We give a more complete survey in section 2.

for adopting the strong definition. First, it is essential for some applications that only authenticated content be received. Consider a stream that delivers stock quotes in real time. While it might be acceptable to lose a quote, we must ensure that only authenticated quotes are ever displayed. Secondly, our constructions which resist loss in the strong sense can easily be adapted to the weaker notion of resistance.

Existing authentication schemes that resist packet loss have been designed to resist *worst-case* packet loss. Any number of packets may be lost anywhere in the sequence, without interfering with the receiver's ability to authenticate the packets that arrived. Studies conducted on packet loss in UDP suggest that resisting worst-case packet loss is an overkill. The focus should be instead on resisting *random* packet loss. We will show how that leads to much more efficient constructions.

Since packet loss on the network is not malicious, it is natural to analyze the patterns of loss and design our authentication schemes accordingly. In [6], Paxson shows that on the Internet consecutive packets tend to get lost together in a *burst*. We adopt this model and propose authentication schemes designed to resist bursty loss. Specifically, our goal is to maximize the size of the longest single burst of loss that our authenticated streams can withstand. Of course, this is not to say that our constructions resist only a single burst. As will be clear, once a few packets have been received after a burst, our scheme recovers and is ready to maintain authentication even if further loss occurs.

In the next section, we will give a brief overview of related work. In section 3, we present our authentication scheme, which is constructed to take advantage of the burst model of packet loss suggested by Paxson. We show that our solution is efficient both in terms of computation and communication overhead. In section 4 we prove that our scheme is optimal given certain constraints. In section 5, we argue that the model used to evaluate our authentication scheme is robust, in the sense that our schemes remain close to optimal even under slightly different assumptions. Finally, we discuss the implementation of our constructions in the last section.

## 2. Related work

The computational bottleneck of an authentication scheme is the signing operation. Digital signatures are expensive to generate and verify. As a rule of thumb, a desktop making full use of its CPU can process on the order of 100 DSA signatures [5] per second. There are two complementary approaches to improving efficiency: designing faster signature schemes and amortizing each signing operation by making use of a single digital signature to authenticate several packets. We will review in turn these two approaches.

A variety of generic techniques exist for speeding up signatures. For example, a small public key reduces work for the verifier. The Chinese Remainder Theorem makes fast "divide and conquer" computations possible. Finally, time/memory trade-offs are possible with precomputation. Using a combination of these techniques, Wong and Lam propose in [9] an optimized version of the Feige-Fiat-Shamir signature scheme [2, 3]. This optimization achieves verification rates comparable to RSA with small exponent, while signing is twice as fast as DSA. The same paper introduces "adjustable and incremental" signatures, i.e. signatures that can be verified at different levels of confidence depending on the resources available to the verifier.

One-time (or k-time) signature schemes (OTS) offer a significant speed-up over regular signatures. In [4], Gennaro and Rohatgi propose a hybrid scheme, in which a single digital signature is used to initiate a chain of OTS. The drawback of OTS is that their size is proportional to the number of bits of the quantity being signed. A OTS computed on a message hashed with SHA-1 is on the order of 1000 bytes. An approach to make OTS smaller is to reduce the size of the message hash. In [8], Rohatgi shows how this can be done without reducing security. The idea is to hash the message with a family of Target Collision Resistant (TCR) hash functions. A TCR family of $2^{40}$ keyed 80-bit hash functions offers with only 120 bits the same security as a single 160-bit hash function, because a birthday attack on the family is $2^{40}$ times harder. The scheme proposed in [8] achieves 300 bytes per signature for 1000 signatures per second.

Let us now turn to solutions which amortize each digital signature over several packets. In [9], Wong and Lam propose buffering packets into groups. A single signature is computed for each group on some function $f$ of all the hashes of the packets. In addition to this common signature, each packet carries some ancillary information from which the value $f$ can be recovered. This allows each packet to be verified independently of the others. In the simplest setting, the hashes of all the packets in the group are concatenated into a string, and $f$ is a hash of this string. The ancillary information consists of the hashes of all the other packets. A variant uses a binary tree to reduce the size of the ancillary information. The hashes of packets are placed at the leaves of the tree, while inner nodes contain a hash of the concatenated values of their children. To be verifiable, each packet need only include the values of the siblings of the nodes along the path to the root.

Another variation on the same scheme uses the efficient k-time signature of [8] which was introduced above. Instead of a hash, we sign a public-key for a k-time signature scheme at the root of the tree. An instance of this signature is used for each packet. The communication overhead per packet is higher, but unlike the proposal of Wong and Lam, this variant does not require to buffer packets into groups before sending them. The original scheme and these variants are reasonably fast and resistant to packet loss in the worst-case. There is a trade-off between efficiency (many packets per group) and communication overhead (more ancillary information).

Finally, Perrig et all propose in [7] two efficient solutions to the problem of authenticating streams in a lossy environment. In the first scheme (TESLA,) packets are authenticated with MACs. The MAC keys are disclosed after a certain time interval, to enable verification. The delay before disclosure is chosen long enough to ensure that the keys can no longer be used to corrupt packets. TESLA is highly efficient and versatile, but it requires the ability to synchronize the clocks of the sender and the receiver within some margin. The second scheme (EMSS,) uses one-way hashes in combination with digital signatures to achieve authentication, following an idea introduced in [4]. To resist loss, the hash of each packet is stored in multiple locations (we use a similar strategy.) EMSS proposes to choose these locations at random, and provides probabilistic guarantees that a packet can be authenticated given a certain amount of loss in the stream. In contrast, our constructions are deterministic (thus possibly easier to implement,) and optimized to resist bursty loss.

## 3. Our scheme

We consider a stream exchanged between a sender and a receiver over an insecure, unreliable channel such as UDP. Lost packets are not retransmitted, and packets may arrive out of order. We make the assumption that loss occurs in bursts. A burst starts at a location randomly chosen in the sequence and lasts for a randomly chosen number of packets.

Our approach to signing the stream follows an idea introduced in [4]. We use a combination of one-way hashes and digital signatures to authenticate packets. The idea is as follows: if a collision-resistant hash of packet $P_1$ is appended to packet $P_2$ before signing $P_2$, then the signature on $P_2$ guarantees the authenticity of both $P_1$ and $P_2$.

More generally, a packet $P_0$ can be authenticated as long as there exists a sequence of packets $P_1, ..., P_n$ such that: the hash of $P_0$ is appended to $P_1$, whose hash is in turn appended to $P_2$, etc, and the last packet $P_n$ is signed.

We divide the stream into sequences of packets. A sequence may consist of anywhere between a few hundred and a few thousand packets. From now on, we restrict ourselves to the problem of authenticating one sequence, the last packet of which is signed. We study where to append the hashes of packets within the sequence to provide optimal resistance to bursty loss.

Throughout the paper, we make use of the following notations:

- $p$: the number of packets buffered on the sender side before transmission. For a stream broadcast strictly in real-time, $p = 1$. In most applications we expect some buffering, and thus $p$ is usually a small integer.
- $h$: the capacity of the buffer that the sender uses to store packet hashes. $\tilde{h}$ is the average number of items present in the buffer.
- $m$: the maximum number of hashes that may be appended to a packet. $\tilde{m}$ is the average number of hashes appended to a packet.

Likewise, we consider that the receiver has a packet buffer and a hash buffer to process the stream. On the receiver side however, the packet buffer only serves to restore the order of the packets as they arrive. A packet is considered lost if it is not received by the time the buffer is full. We thus ignore the packet buffer for the receiver and assume that a packet is either considered lost or arrives in order.

We represent a sequence of $n$ packets $P_1, ..., P_n$ by a directed acyclic graph. Only the last node $P_n$ is signed. 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 graph may not contain a cycle (so in particular $i \neq j$), but there is no requirement that $i < j$. The sequence is *fully authenticable* if there exists at least one directed path from every node to the last signed node.

We are not concerned with the possible loss of the signature packet. For one thing, if the number of packets per sequence is large enough, it is highly unlikely that the signature packet will be lost. In any case, it is always possible to transmit multiple copies of the signature.

Two parties must agree on an authentication scheme to exchange sequences of any length. This scheme specifies where hashes should be appended in a sequence of $n$ packets. Using our graph representation, we define a *scheme* $S$ as a function: $n \to S(n)$ where $S(n)$ is a directed acyclic graph on a set of $n$ nodes, or more. (We allow padding with dummy packets.)

We require that the sequence $S(n)$ be generated and verified in the following way. The sender computes all the hashes that need to be appended to the first packet to be sent. This may trigger recursive hash computations. For example, if the hashes of packets 2 and 5 must be appended to packet 1, the sender has to compute these hashes first. This can not be done until the hashes that must be appended to packets 2 and 5 respectively have themselves been computed. Since there are no cycles in the graph, these recursive calls eventually come to an end. The first packet is then ready to be sent, and the operation is repeated with the next packet. Of course, packet hashes are computed only once, and then buffered in memory until they are no longer needed.

The receiver verifies a sequence in much the same way. Each time a packet arrives, the hashes found appended to it are buffered. They are kept in memory until the packet they authenticate has arrived or been lost. We define:

• $h'_{Loss}$: the buffer capacity neeeded by the receiver to validate any sequence of packets.

• $h'_{OK}$: the capacity sufficient to validate sequences for which no packet was lost.

A scheme $S$ is *periodic* of period $s > 0$ if the following two properties hold:

• The function $S$ is piece-wise constant over intervals of length $s$.

• There exists $N > 0$ such that for all $n \geq N$, $S(n + s)$ is obtained by prepending to $S(n)$ itself the first $s$ nodes of $S(n)$ along with the edges coming out of them (see Figure 1).

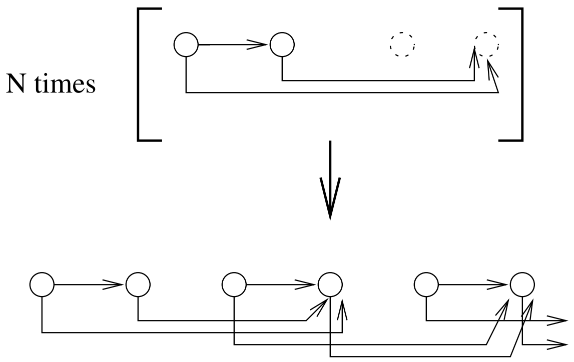


Figure 1. A periodic scheme of period 2

Finally, we define $B(P_i)$ as the largest integer $j$ such that $\forall 1 \leq k \leq j$ a burst of length $k$ starting at packet $P_{i+1}$ leaves the rest of the sequence $(G - \{P_{i+1}, ..., P_{i+k}\})$ fully authenticable.

We extend this definition to a periodic scheme $S$:

• $B_S = min_i B(P_i)$ where the minimum is taken over a period of $S$.

• $\tilde{B}_S = \frac{1}{n} \sum B(P_i)$ where the average is taken over a period of $S$.

### 3.1. Our solution in the case $p = 1$ (no packet buffering on the server side)

This simple case is of practical importance and introduces the basic building block for our generic construction. We propose a family of schemes $C_a$, parametrized by the integer variable $a$. $C_a$ is a periodic authentication scheme of period 1 defined as follows: the hash of packet $P_i \in C_a$ is appended to two other packets: $P_{i+1}$ and $P_{i+a}$. The last node $P_n$ is signed. We call $C_a$ a chain of strength $a$.

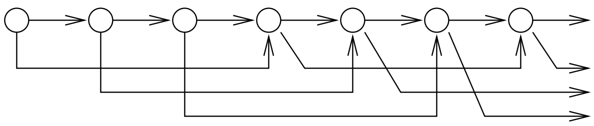

Figure 2. The authentication chain $C_3$

**Characteristics of these schemes:** This family of schemes is well-defined: all the $C_a(n)$ are acyclic. The maximum number of hashes appended to any packet (denoted $m$), and the average number of hashes appended to a packet ($\tilde{m}$) are both equal to 2.

**Sender:** Chain $C_a$ can be executed by a sender who buffers $p = 1$ packet and has a hash buffer of capacity $h = a$.

**Receiver:** If no packets are lost, the hash of packet $P_i$ can always be verified when $P_{i+1}$ arrives and then discarded. Therefore $h'_{OK} = 1$. After a burst of loss, the only extra hash to remember is that of the packet immediately preceding the burst, and so $h'_{Loss} = 2$.

**Resistance to Loss:** $B_{C_a} = \tilde{B}_{C_a} = a - 1$

It is not hard to convince oneself that bursts of length up to $a - 1$ do not disconnect any packet from the signature. We will prove in the next section that chain $C_a$ is optimally resistant to bursty loss among all the schemes that can be executed by a sender who buffers 1 packet and has a hash buffer of size $h = a$. Intuitively, $C_a$ resists loss because it stores packet hashes in locations as far apart as the size of the sender hash buffer allows.

### 3.2. The generic construction for $p > 1$

When the sender buffers $p > 1$ packets, it becomes possible to append the hash of a packet to one that precedes it. In fact, with a buffer of size $p$, we may append to a packet any of the hashes of the next $p - 1$ packets.

The constructions we propose are extensions of chains. We introduce $p - 1$ additional packets between those of the original chain $C_a$ to create *augmented chain* $C_{a,p}$. We will soon discuss how to link these new packets to integrate them in the authenticated sequence.

We start with a simple example (see Figure 3) to show how to augment chain $C_3$ when $p = 2$ or $p = 3$. We number the newly inserted packets with integers, and use letters for the packets of the original chain. The drawing at the top of figure 3 corresponds to the case $p = 2$ (a single additional packet between packets of the original chain.) We have represented both the newly inserted edges, and those belonging to the original chain. For $p = 3$ (two new packets between those of the original chain,) we have only shown the new packets (1 and 2) and the new edges to be inserted between the packets of the original chain.

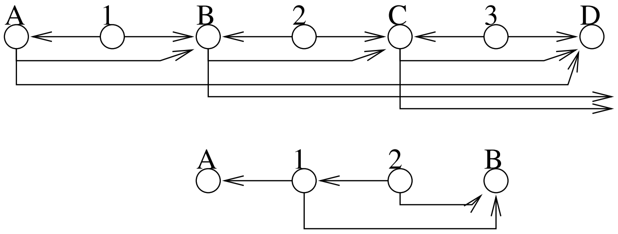


Figure 3. Augmented chain $C_{3,p}$ with $p = 2$ (on top) and $p = 3$ (below)

Let us now consider the general case. We propose two ways of inserting new packets, which are equally resistant to bursty packet loss. Our first structure is described in figure 4. The hash of each new packet is appended both to the packet preceding it and to the packet from the original chain following it. This structure is very easy to implement, but has the drawback that the maximum number of hashes appended to a packet grows linearly with $p$ (although observe that the average number of hashes appended is $\tilde{m} = 2$). This simple structure is well suited for small values of $p$.

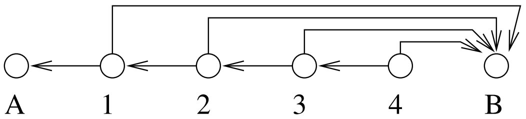


Figure 4. A simple way of inserting additional packets

We now present a more complex structure for which $m$ is constant. This is our main scheme. We refer to it as *augmented chains* and will analyze it in detail below.

We define recursively how to insert an (even) number of new packets in the original chain. The starting point is the structure proposed in the bottom part of Figure 3, which indicates how to insert two new packets (1 and 2) between the packets of the original chain ($A$ and $B$.)

Now to insert 4 packets between $A$ and $B$, we proceed recursively (see Figure 5.) We first insert two new packets 1 and 2 as above, then insert two more packets 3 and 4 between 1 and 2 in exactly the same way that 1 and 2 were inserted between $A$ and $B$. This process is generalized in a straightforward way to insert any even number of new packets.

**Characteristics of augmented chains:** The family of augmented chains is well-defined: all the $C_{a,p}(n)$ are acyclic. The maximum number of hashes appended to any

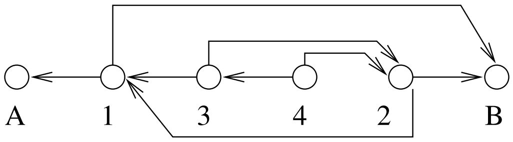


Figure 5. Recursive insertions for augmented chains

packet is constant ($m = 5$), and the average number of hashes appended to a packet is $\tilde{m} = 2$.

**Sender:** Augmented chain $C_{a,p}$ can be executed by a sender who buffers $p$ packets and has a hash buffer of capacity $h = a + p - 1$. Indeed, the hashes of the $p-1$ inner packets must now be stored in addition to the $a$ hashes corresponding to packets in the main chain.

**Receiver:** $h'_{OK} = \frac{p+3}{2}$ and $h'_{Loss} = h'_{OK} + 2$. Indeed, let us consider the impact of the $p-1$ new packets for the receiver. The first $\frac{p+1}{2}$ of those are linked together and can be processed at the cost of remembering one extra hash only. But they also carry the hashes of the $\frac{p-3}{2}$ last inner packets, bringing the total number of hashes that must be stored in the buffer to $\frac{p+3}{2}$.

**Resistance to loss:** $B = p(a-1)$
The first step is to observe that packets newly inserted between two packets of the original chain are reachable from either of these chain packets. Now let us consider a burst of loss. Let $C$ and $D$ be the packets in the outer chain immediately preceding and succeeding the burst. All the packets between $C$ and the start of the burst are reachable from $C$. Similarly, all the packets between the end of the burst and $D$ are reachable from $D$. Thus the whole chain remains authenticable as long as $C$ is connected in the outer chain to a packet beyond $D$. This is always the case if $B \leq p(a-1)$. In the next section we prove that this value of $B$ is optimal.

To parametrize our construction on the resources available to the sender, we can rewrite the equation as $B = p(h-p)$. Recall that $p$ and $h$ are respectively the sizes of the packet and the hash buffer on the sender size. If the server can distribute available memory between a buffer for hashes and a buffer for packets, $B$ is maximal when the memory is equally allocated between these two buffers. In practice, if a hash is 20 bytes and a packet 1000 bytes, one would expect $h \approx 50p$. Of course, other considerations might come into play when deciding on the respective sizes of the hash and the packet buffer.

## 3.3. Comparison with other schemes

We compare our scheme with those proposed by Wong and Lam in [9]. Recall that the schemes of [9] come in three basic flavors, depending on how packets are organized into groups. Figure 6 summarizes these three options.

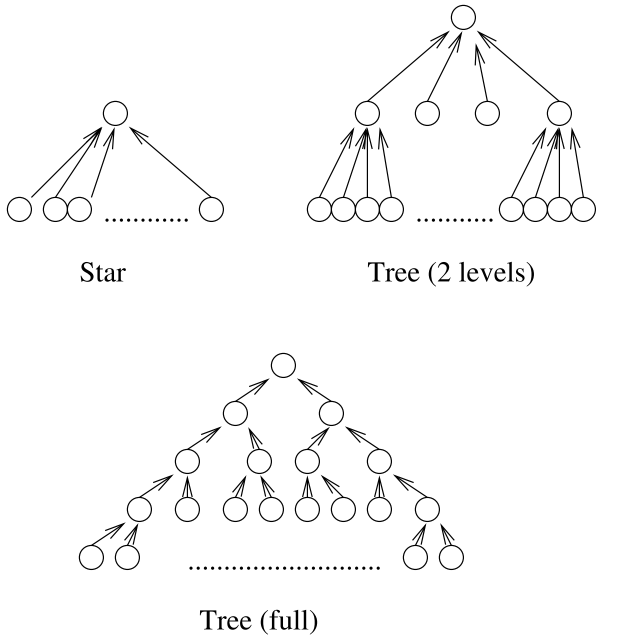


Figure 6. The schemes of Wong and Lam

For our comparison, we consider a stream divided into groups (called sequences in our scheme) of 16 packets. A single digital signature is generated by the sender and verified by the receiver for each group. We use the following measures to compare our scheme with others:

• Hash: the total number of hashes computed by the sender(The number is the same for the receiver.)

• Overhead: the overhead per packet in bytes.

• Loss: the type of loss that the scheme resists.

• Delay: the delay on the receiver side (in number of packets) before authentication is possible.

| Scheme | Hash | Overhead | Loss | Delay |
|---|---|---|---|---|
| WL star | 17 | 340 | any | 0 |
| WL tree(2 levels) | 21 | 160 | any | 0 |
| WL tree(full) | 31 | 120 | any | 0 |
| Augmented chain | 16 | 43 | bursts | 16 |

**Applications**

The family of augmented chains is a highly efficient authentication scheme for streams, with obvious applications in settings where computational and communication resources are limited, and where there is no guarantee that all packets will be delivered.

Even where network bandwidth is not scarce, the low communication overhead possible with our scheme may be crucially important for the following reason. Short of opening a special communication channel for authentication (a costly solution,) authentication data must be embedded within the stream itself, conforming to protocols for sending and receiving streams that were not designed to allow authentication. A number of techniques for embedding authentication data in the stream are described in [4]: water-marking, use of a USER-DATA section in MPEG audio or video, etc... These techniques offer either very little space, or offer space at the cost of degrading the quality of the data.

We expect that the low computation and communication overhead of our schemes will make them useful in a variety of applications.

## 4. Proof of optimality

We show here that the constructions of the previous section offer optimal resistance to bursty packet loss for authenticated streams, given the resources available to the sender and the receiver.

Let us start with a simple observation. If authentication is to be possible when packets may get lost, the hash of each packet must be stored in at least two distinct locations inside the stream. This implies that the average number of hashes appended to each packet can not be less than 2. Our scheme achieves this lower bound ($\tilde{m} = 2$.)

We now turn to the proof that augmented chains are optimally resistant to bursty packet loss given the resources allocated to the sender. We must first introduce some notations. We define the *scope* of a packet $P_i$ with the following two variables:

• Forward scope: $f(P_i)$ is the maximum of $j-i$ over all indices $j > i$ for which there exists a directed *edge* from $P_i$ to $P_j$. If there is no such $j$, set $f(P_i) = 0$.

• Backward scope: $b(P_i)$ is the maximum of $i - j$ over all indices $j < i$ for which there exists a directed *path* from $P(i)$ to $P(j)$. If there is no such $j$, set $b(P_i) = 0$. Observe that a periodic scheme $S$ is executable by a sender who buffers $p$ packets if and only if for all $n$, $b(P_i) < p$ for all $P_i \in S(n)$.

A periodic scheme $S$ is executable by a sender with a hash buffer of capacity $h$ (resp of average capacity $\tilde{h}$) if and only if for all $n$ and for all $P_i \in S(n)$, there are $\leq h$ nodes (resp. on average $\tilde{h}$ nodes) $P_j$ for which either:

• $j < i$ and $f(P_j) \geq i - j$.

• $j \geq i$ and $b(P_j) \geq j - i$.

Indeed, the hash of packet $P_j$ must be present in the hash buffer over the interval $[P_{j-b(P_j)}; P_{j+f(P_j)}]$. The condition expresses that the buffer may contain at most $h$ hashes at node $P_i$.

**Lemma 4.1**

$$B(P_i) < \max_{0 \leq j \leq b(P_i)} [f(P_{i-j}) - j]$$

*In particular for a node without back-edges (that is, $b(P_i) = 0$), $B(P_i) < f(P_i)$*

**Proof** Let $m = \max_{0 \le j \le b(P_i)}[f(P_{i-j}) - j]$. Now consider the interval $I = [P_{i+1}; P_{i+m}]$. We show that any directed path from $P_i$ to the signature $P_n$ goes through at least one node of $I$. Thus the disappearance of $I$ in a burst of length $m$ leaves the signature unreachable from $P_i$ which proves the lemma.
Now let $A_1 \dots A_r$ be a path from $P_i$ to $P_n$. By definition of $b(P_i)$, for all $j$ we have $A_j \in [P_{i-b(P_i)}; P_n]$. Let $s$ be the maximum index such that for all $1 \le t \le s$ we have $A_t \in [P_{i-b(P_i)}; P_i]$. $A_s = P_{s'}$ with $i - b(P_i) \le s' \le i$. Then $A_{s+1} \in [P_{i+1}; P_{s'+f(P_{s'})}] \subset I$.

### 4.1. Optimal $B$ with constraints on $h$ and $p$

We have shown that augmented chain $C_{a,p}$ can sustain a burst of length up to $p(a-1)$. The following proposition shows that this is the maximum possible for a scheme that can be executed by a sender who buffers $p$ packets and has a hash buffer of capacity $a + p - 1$.

**Proposition 4.2** *Let $S$ be an authentication scheme that can be executed by a sender who buffers p packets and has a hash buffer of capacity h. Then if $p > (h+1)/2$ we have $B_S \le \left(\frac{h+1}{2}\right)^2$, and if $p < (h+1)/2$ we have $B_S \le p \cdot (h + 1 - p)$*

**Proof** Let $n$ be the period of $S$. Let us consider a sequence of nodes $P_1, P_2, P_3, \dots$ We consider the subsequence of nodes which have no back-edges: $P_{a_1}, P_{a_2}, P_{a_3}, \dots$ Observe that $a_i - a_{i-1} \le p$ since the sender buffers at most $p$ packets. Now let $a_m$ be the index (or one of the indexes if there are several) for which $d_m = a_{m+1} - a_m$ is maximal.
We consider the $h + 1 - d_m$ nodes without back-edges preceding $P_{a_m}$. By lemma 4.1

$$B_S < min_{1 \le i \le h+1-d_m} f(P_{a_{m-i}})$$

At least one of those nodes cannot have any forward edge extending beyond $P_{a_m}$, for otherwise the hash buffer would contain $> h$ hashes at point $P_{a_m}$.
Therefore

$$B_S < \max_{1 \le i \le h+1-d_m} (a_m - a_{m-i})$$

Since for all $i$, $a_m - a_{m-i} \le i.d_m$, $B_S \le d_m(h+1-d_m)$
If we consider the expression above as a function of $d_m$, the maximum is obtained for $d_m = (h+1)/2$. But remember that we also require $d_m \le p$
So if $p \ge (h+1)/2$ we have $B_S \le \left(\frac{h+1}{2}\right)^2$, and if $p < (h+1)/2$ we have $B_S \le p \cdot (h + 1 - p)$

### 4.2. Structure of optimal schemes.

In fact, the proof of Proposition 4.2 reveals the structure that a scheme must have in order to maximize $B$. The following definition will help expose this structure. We say that a directed acyclic graph on $p$ nodes $P_1, \dots, P_p$ has the *extremity property* if for all $P_i$ ($i \ne 1, p$) the following two conditions hold:

- There exists a directed path from $P_i$ to $P_1$ included in the interval $[P_1; P_i]$
- There exists a directed path form $P_i$ to $P_p$ included in the interval $[P_i; P_p]$

**Proposition 4.3** *Let $S$ be an authentication scheme which can be executed by a sender who buffers p packets and has a hash buffer of size h. If $B_S$ is maximal, then $S(n)$ has the following structure:*

- *Nodes without back-edges are regularly spaced, at intervals of p nodes.*
- *The subgraph of $S(n)$ between two consecutive nodes without back-edges has the extremity property.*

**Proof** Follows directly from the proof of proposition 4.2.

## 5. Alternate models

In this section we argue that our model for stream authentication is robust, in the sense that our constructions remain close to optimally resistant to bursty packet loss under slightly different assumptions.

We study first what happens if we constrain the *average* capacity (rather than the *maximum* capacity) of the buffer available to the sender to store hashes. For a sender servicing several clients in parallel, the average memory requirement of each connection over time might be a more meaningful measure than the maximum capacity required by each connection. We prove in section 5.1 that the longest burst a sequence can sustain in this setting is essentially the same.

In section 5.2, we consider the problem of maximizing $\tilde{B}$, the length of the average longest burst of loss that an authenticated sequence can sustain (here, the average is taken over the locations where the burst may start.) Assuming that network loss is not malicious, it makes sense to maximize the longest burst that can be sustained on average. We prove that the longest *average* burst a sequence can sustain is close to the longest *worst-case* burst, and that our constructions remain close to optimally resistant to bursty packet loss.

### 5.1. Hash buffer of average capacity.

**Proposition 5.1** *Let $S$ be a scheme that can be executed by a sender who buffers p packets and has a hash buffer of* average *capacity $\tilde{h}$. If $p \le \tilde{h} - 3/2$, we have $B_S \le p(\tilde{h} - \frac{p+3}{2})$. If $p > \tilde{h} - 3/2$, we have $B_S \le \frac{1}{2}\left(\tilde{h} - \frac{3}{2}\right)^2$.*

This result should be compared with Proposition 4.2. The bounds we get when we constrain the maximum capacity of the hash buffer ($B_S \leq p(h+1-p)$) and when we constrain the average capacity ($B_S \leq p(\tilde{h} - \frac{p+3}{2})$) are on a similar order of magnitude. We start the proof of Proposition 5.1 with the following lemma:

**Lemma 5.2** *Let $S$ be a scheme of period $s$, that can be executed by a sender who buffers p packets and has a hash buffer of average capacity $\tilde{h}$. Let $P_1, ..., P_s$ be any sequence of $s$ consecutive nodes of $S(n)$. We have* $\sum_{i=1}^{s}(f(P_i) + b(P_i) + 1) \leq \tilde{h}s$

**Proof** In section 4 we proved that the hash of $P_i$ must be present in the hash buffer over the interval $[P_{i-b(P_i)}; P_{i+f(P_i)}]$ which is of length $f(P_i) + b(P_i) + 1$. Taking the average over a period of the scheme, we get the lemma.

**Proof** (proposition 5.1) Let $n$ be the period of $S$. We consider $n$ consecutive nodes $P_1, ..., P_n$. At least $n/p$ of those nodes have no back-edges, say $P_{a_1}, \ldots, P_{a_k}$. Now by lemma 4.1:

$$B_S < min_{1 \leq i \leq k} f(P_{a_i})$$

Since by lemma 5.2

$$\sum_{i=1}^{n}(f(P_i) + b(P_i) + 1) \leq \tilde{h}n$$

We have

$$\sum_{i=1}^{k} f(P_{a_i}) \leq (\tilde{h}-1)n - \sum_{i=1}^{n} b(P_i) - \sum_{P \neq P_{a_1}, \ldots, P_{a_k}} f(P)$$

But for all $P$, $f(P) \geq 1$, and so

$$B_S < \frac{1}{k}\left((\tilde{h}-1)n - \sum_{i=1}^{n} b(P_i) - (n-k)\right)$$

We must now give a lower bound for $\sum_{i=1}^{n} b(P_i)$. For any node $P_i$ between $P_{a_j}$ and $P_{a_{j+1}}$, $b(P_i) \geq i - a_j$. Taking the sum over all nodes between $P_{a_j}$ and $P_{a_{j+1}}$ gives $\sum b(P_i) \geq 1/2(a_{j+1} - a_j)(a_{j+1} - a_j - 1)$. Finally summing over all intervals, we get

$$\sum_{i=1}^{n} b(P_i) \geq \sum_{j=1}^{k} 1/2(a_{j+1} - a_j)(a_{j+1} - a_j - 1)$$

But $\sum_{j=1}^{k}(a_{j+1} - a_j) = n$ and therefore:

$$\sum_{i=1}^{n} b(P_i) \geq k \cdot \frac{1}{2} \cdot \left(\frac{n}{k}\right)\left(\frac{n}{k} - 1\right)$$

So finally

$$B_S < \frac{1}{k}\left((\tilde{h}-1).n - \frac{n}{2k}(n-k) - (n-k)\right)$$

If we consider the expression above as a function of $k$, the maximum is obtained for $k = \frac{n}{\tilde{h} - \frac{3}{2}}$. But remember that we also require $k \geq n/p$.

So if $p > \tilde{h} - 3/2$ we have $B_S \leq \frac{1}{2}\left(\tilde{h} - \frac{3}{2}\right)^2$, and if $p \leq \tilde{h} - 3/2$ we have $B_S \leq p(\tilde{h} - \frac{p+3}{2})$

## 5.2. Optimal $\tilde{B}$

**Proposition 5.3** *Let $S$ be a scheme that can be executed by a sender who buffers p packets and can store an average of $\tilde{h}$ hashes in memory. Then $\tilde{B}_S < (\tilde{h} - 1)p$*

This result should be compared with Proposition 4.2 and Proposition 5.1. It turns out that the optimal value for $\tilde{B}$ is not far from the optimal value for $B$.

**Proof** Let $n$ be the period of $S$. We consider $n$ consecutive packets $P_1, ..., P_n$. Necessarily $b(P_i) < p$ and so by lemma 4.1,

$$B(P_i) < \max_{0 \leq j \leq p-1} [f(P_{i-j}) - j]$$

So

$$\tilde{B}_S = \frac{1}{n}\sum_{i=1}^{n} B(P_i) \leq \frac{1}{n}\sum_{i=1}^{n}\left(\max_{0 \leq j \leq p-1}[f(P_{i-j}) - j] - 1\right)$$

$$\tilde{B}_S < \frac{1}{n}\sum_{i=1}^{n}\sum_{j=0}^{p-1} f(P_{i-j}) = \frac{1}{n}\sum_{j=0}^{p-1}\sum_{i=1}^{n} f(P_{i-j})$$

Since by lemma 5.2 $\sum_{i=1}^{n} f(P_i) \leq (\tilde{h}-1)n$, we have $\tilde{B}_S < \frac{1}{n}\sum_{j=0}^{p-1}(\tilde{h}-1)n = (\tilde{h}-1)p$

# 6. Implementation

We have implemented our constructions as plug-ins to the *RealSystem* platform from Real Networks [10] to authenticate audio and video streams.

RealSystem consists of a streaming server and many client RealAudio players. The server itself consists of a core and many supporting modules which are responsible for reading files, packetizing data, adding transport headers and so on.

Our implementation replaces the file-format plug-in. This plug-in is responsible for providing the server core with packetized data that contain authentication information.

The file-format plug-in can be controlled through a configuration file. This file specifies how often signatures are computed and for testing purposes, how often a burst of packet loss occurs. In our example configuration we set $p = 7$ (the number of packets buffered on the sender side,) and computed signatures every 49 packets. Figure 7 shows the state of a player after a signature has verified. Figure 8 shows the state after a signature verification fails.


Figure 7. Verified signature

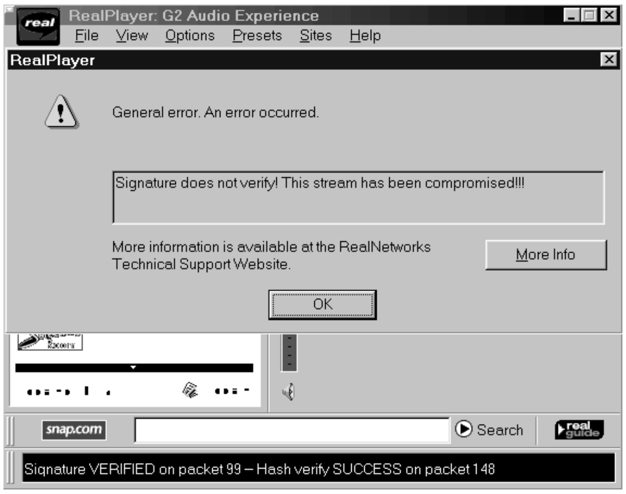


Figure 8. Signature verification failed

In order to ensure that authenticated streams are appropriately associated with our rendering plug-in, we created a new mime-type. We append the extension ".apf" to the original filename and associate our client plug-in with this mime-type. For example the server delivers the file "demo.rm" with authentication information when the client requests "demo.rm.apf". Appending our own extension allows our plug-in to dynamically determine the original rendering plug-in for the requested stream.

It should be noted that without our plug-in a player is unable to play streams with embedded authentication information.

The plug-ins and source code are available for download from [11].

## 7. Conclusion

We propose a new stream authentication scheme. In contrast to existing solutions, our scheme resists random packet loss rather than worst-case packet loss. We prove that our construction is optimally resistant to bursty packet loss given the resources available to the sender and the receiver, and has the lowest possible communication overhead.

## Acknowledgments

Since the beginning of their work on this paper, the authors are grateful to Dan Boneh for discussions and numerous helpful comments. We would like to thank the anonymous reviewers for their suggestions to improve the exposition of this paper.

## References

[1] S. Even, O. Goldreich and S. Micali. On-line/Off-line Digital Signatures. In *Journal of Cryptology*, Volume 9, Number 1, Winter 1996.

[2] U. Feige, A. Fiat and A. Shamir. Zero Knowledge Proofs of Identity. In *Proc. of the 19th Annual ACM Symposium on Theory of Computing*, 1987.

[3] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology - CRYPTO '86*, pages 186-194. Lecture Notes in Computer Science 263, Springer-Verlag, 1986.

[4] R. Gennaro and P. Rohatgi. How to Sign Digital Streams. In *Advances in Cryptology - CRYPTO '97*, pages 180-197. Lecture Notes in Computer Science 1294, Springer-Verlag, 1997.

[5] National Institute of Standards and Technology. Digital Signature Standard. NIST FIPS PUB 86, U.S. Department of Commerce, May 1994.

[6] V. Paxson. End-to-End Internet Packet Dynamics. *IEEE/ACM Transactions on Networking,* 7(3):277-292, June 1999.

[7] A. Perrig, R. Canetti, J.D. Tygar and D. Song. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. In *Proc. of IEEE Security and Privacy Symposium*, May 2000.

[8] P. Rohatgi. A Compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication. In 6*th ACM Conference on Computer and Communication Security*, November 1999.

[9] C.K. Wong and S.S. Lam. Digital Signatures for Flows and Multicasts. In *Proc. IEEE ICNP '98*, October 1998.

[10] http://www.realnetworks.com

[11] http://crypto.stanford.edu/ nagendra/projects/StreamAuth/StreamAuth.html