# Authenticating Real Time Packet Streams and Multicasts

Alain Pannetrat, Réfik Molva
Institut Eurécom, Sophia-Antipolis, France.

## Abstract

*In this work we propose a new stream authentication scheme that is suitable for* live *packet streams distributed over a lossy channel, such as an IP-multicast group. Packets are signed together in a block and the recipient can authenticate this block if the loss rate per block is lower than a certain threshold, which can be chosen based on the characteristic of the communication channel. This scheme provides both integrity and non repudiation of origin, and in a majority of situations, it performs with less overhead in bytes per packet than previously proposed practical* live *stream authentication schemes.*

## 1. Introduction

Authentication is a primary requirement for the delivery of multicast content such as live television video, radio broadcastings, or stock quotes over the Internet. The recipients of the content will often require guarantees of integrity and sometimes even non-repudiation on the received data. Conversely, the content provider does not want to be impersonated by another party. The two major challenges in the design of a multicast authentication mechanism are first, the *best effort* characteristic of the communication channel and second, the *live* characteristic of the distributed content. A lot of Internet video or audio streaming protocols are designed to tolerate some packet loss patterns with a graceful degradation in playback quality. A good authentication scheme must allow the recipient to authenticate the received data despite these losses. While in some examples, such as a pre-recorded video, it is possible to compute the authentication information to be sent with the packets *offline*, in many cases such as *live* or real time event broadcasts, the computations need to be done *online* or with reasonably small buffering. Interesting protocols targeted mainly for *offline* streams have been proposed, for example, by Miner and Staddon[9], were the sender is assumed to buffer very large amounts of data. Our goal is to provide a scheme which works with the constraints of a *live* event broadcast, for streams that are not known in advance.

A desirable property of a live authentication stream is to allow a receiver to start authenticating from any point in the stream. Though we would ideally like to be able to start authentication on a packet boundary, in practical situations we believe that starting authentication on the boundary of a group of packets is a satisfactory compromise. To describe this feature, we will say that a stream authentication scheme is *joinable* on a certain boundary.

A straitforward stream authentication method would be to use a digital signature on each packet of the stream. But this approach has serious drawbacks. First, the computational cost of signing packets individually is prohibitive in many scenarios. Second, adding a typical 1024 bit signature[16] (or 128 bytes) to every packet represents a really high overhead. Faster digital signatures designed for stream authentication where proposed in [15] and [17], but these solutions have an even higher overhead, and today, signing each packet remains impractical.

An alternative approach is to amortize the signature over several packets in a *block*. The stream is itself divided into many small blocks that have each a unique signature that is combined with hash techniques to authenticate the packets in the block. We refer to these techniques as well as the one we propose in this work as "hybrid" approaches, and we review them is section 5.

The central contribution of this work is the proposal of an original *joinable live lossy* stream authentication scheme with non-repudiation of origin. It uses Erasure Codes[14] to provide a lower overhead per packet than previous live authentication stream proposals, while being adapted to realistic multicast Internet loss patterns.

The next section will introduce a few notations that we will use throughout this work. Our scheme is formalized in section 3 as well its relationship with Internet loss patterns based on a Markov chain model. Section 4 discusses the cost and overhead of our scheme and presents its use in a few concrete scenarios. Finally, we review other live lossy stream authentications schemes in section 5 and compare them with our approach.

Due to space restrictions, some details and proofs were omitted or condensed in this paper. They can however be found in the full version of this work [11].

## 2. Background

**Erasure Codes.** A systematic erasure code algorithm $C_{k,r}$ takes a set $X = \{x_1, ..., x_k\}$ of $k$ source packets and produces a set $\{x_1, ..., x_k, y_1, ..., y_r\}$ of code packets. The set $Y = \{y_1, ..., y_r\}$ denotes the set of parity symbols generated by the code and we will often simply write $\{X;Y\} \leftarrow C_{k,r}(X)$ to describe the code generation process. Any subset $Z$ of $k$ elements in $\{X;Y\} = \{x_1, ..., x_k, y_1, ..., y_r\}$ is sufficient to recover the set of source packets $X$, with a corresponding decoding operation denoted $\{X\} \leftarrow D_k(Z)$. For additional details on erasure codes, we refer the reader to [14].

**Notations.** In this work we will consider a stream to be divided in consecutive blocks of $b$ packets. We allow the use of dummy padding null packets at the very end of the stream to match a $b$ packet boundary. Our authentication scheme is parameterized by $b$, the block size in packets, and $p \in [0..1[$, the maximum expected loss rate per block.

We will denote $H$ as a cryptographic hash function such as SHA[10] or MD5[13] which produces hashes of $h$ bytes. The couple $(\mathcal{S}, \mathcal{V})$ will denote the digital signature and verification algorithms respectively associated with the source of the packet stream, such as RSA[16, 1] for example. The size of the signatures will be expressed as $s$ bytes. For RSA, a typical value for $s$ is 128 bytes (or 1024 bits).

## 3. Stream Authentication

### 3.1. Authentication Tags

Consider a block as a sequence of $b$ packets $[P_1, ..., P_b]$. Let $\{h_1, ..., h_b | h_i \leftarrow H(P_i)\}$ be the set of hash values of these packets computed with a hash function $H(\cdot)$. From this hash set we build a set of $b$ authentication tags $Z = \{\tau_1, ..., \tau_b\}$ with the following algorithm $\mathcal{T}_{[b,p]}$ which uses some of the notations introduced in the previous section:

```
Tag generation: $\mathcal{T}_{[b,p]}$
INPUT: $\{h_1, ..., h_b\}$
OUTPUT: $\{\tau_1, ..., \tau_b\}$.
$\{X;\overline{X}\} \leftarrow C_{b,\lceil pb \rceil}(X)$                                  (1)
$\sigma \leftarrow \mathcal{S}(H(h_1||...||h_p))$                                  (2)
$\{Y;\overline{Y}\} \leftarrow C_{\lfloor b(1-p) \rfloor, \lceil pb \rceil}(\overline{X}||\sigma)$   (3)
Split $\{Y;\overline{Y}\}$ into $b$ tags $\{\tau_1, ..., \tau_b\}$.                 (4)
```

To exploit the tag generation algorithm we define our authentication criterion:

**Authentication criterion:** In this work we say that a packet $P_i$ is *fully authenticable* in a block if, given the set of hashes $Z = \{h_1, ..., h_b\}$ of packets in the block and their signature $\sigma = \mathcal{S}(H(Z))$, we can verify that both $\mathcal{V}(\sigma, H(Z)) = true$ and $H(P_i) = h_i$.

The proposed schemes in this work are based on the following property of the tag generation algorithm.

**Proposition 1.** Let $\Pi = [P_1, ..., P_b]$ be a block of $b$ packets and $\{h_1, ..., h_b | h_i \leftarrow H(P_i)\}$ its associated hash set. Given $A = \{\tau_1, ..., \tau_b\} \leftarrow \mathcal{T}_{[b,p]}(\{h_1, ..., h_b\})$, any subset of at least $\lfloor b(1-p) \rfloor$ packets in $\Pi$ can be authenticated using any subset of at least $\lfloor b(1-p) \rfloor$ tags in $A$.

A direct corollary of the proposition above is that both a block of packets and their authentication tags can withstand a loss rate of at most $\lceil pb \rceil$ elements while allowing us to authenticate the remaining packets.

From the construction of the algorithm above we can determine the size of an authentication tag:

**Proposition 2.** Let $h$ define the length of our cryptographic hashes and $s$ the size of the signatures. The size of an individual authentication tag is expressed as a function $\delta(b,p)$ of both the number of packets in a block and $p$ the maximum expected loss rate per block, as:

$$\delta(b,p) = \frac{\mathcal{R}_{\lfloor (1-p)b \rfloor}(s + \lceil p.b \rceil h)}{\lfloor (1-p)b \rfloor}$$

where $\mathcal{R}_n(z)$ is an integer function which returns the lowest multiple of $n$ greater or equal to $z$.

### 3.2. Proposed Schemes

In our stream authentication scheme we propose to piggyback authentication tags in the packets of a block and use *Proposition 1* to authenticate received packets when the loss rate in a block is less than $p$. We propose 3 different variants of our scheme which only differ by the positioning of the authentications tags. In this section we will denote a stream as a set of $m$ blocks $B_1, ..., B_m$. The individual $b$ packets in each block $B_i$ are identified as $P[i,1], ..., P[i,b]$. The corresponding authentication tags are identified as $\tau[i,1], ..., \tau[i,b]$. The packets $P[i,j]$ are a combination of just two things: a stream data packet $D[i,j]$ and an authentication tag.

**ECU: The unbuffered sender scheme.** In this scheme we use packets in a block $B_{(i+1)}$ to piggyback authentication tags related to block $B_i$. The $j^{th}$ packet in a block $B_i$ is thus defined as $P[i,j] = \{D[i,j]||\tau[i-1,j]\}$. This requires the sender to create an extra padding dummy block $B_{(m+1)}$ to allow the last block $B_m$ to be authenticated. This scheme does not require any stream data packet buffering from the sender, only the hashes of the packets in the current block need to be stored by the sender who can then compute the necessary authentication tags to be piggybacked in the next block. In this sense, this scheme is truly a *live* authentication scheme. The tradeoff of this construction is that the receiver will experience a delay of two blocks in the worst case before he can authenticate the first packet in a blocks he received. This construction creates a dependency between two consecutive blocks, thus in the event of a loss that exceeds the threshold $p$ and in particular if a whole block $B_i$ is lost then we will not be able to authenticate $B_{(i-1)}$.

**EC2: The double buffer scheme.** In this scheme, the tags of block $B_i$ are put in packets of the *previous* block $B_{(i-1)}$ and packets in a block $B_i$ are defined as $P[i,j] = \{D[i,j]||\tau[i+1,j]\}$. This requires the sender to create an extra padding dummy block at the beginning of the data stream. The main advantage of this construction is that the receiver can authenticate each received packet immediately upon reception. The main drawback of this scheme is that it requires the sender to buffer two blocks at a time. In this sense it is not a truly *live* scheme but in some applications, this double buffering is still acceptable. Similarly to ECU, this construction also creates a dependency between blocks.

**EC1: The single buffered scheme.** The most obvious construction and perhaps the one that offers the best compromise between the sender buffering and the receiver authentication delay is to piggyback the tags of a block $B_i$ in the block $B_i$ itself. Packets in a block are simply defined as $P[i,j] = \{D[i,j]||\tau[i,j]\}$. This scheme requires the sender to buffer one block and adds a maximum verification delay of one block for the receiver. An advantage of this scheme is that it does not create a dependency between blocks, thus if a block losses packets beyond the expected maximum loss rate $p$, the authentication of neighboring blocks in the stream remains unaffected.

### 3.3. Parameter Choice

Until now we proposed a method which can authenticate a block when a threshold of less than $pb$ packets are lost in a block of $b$ packets. However we need to relate these parameters to concrete average network loss patterns and we will now discuss the choice of the 2 main parameters of our scheme: $b$ the block size and $p$ the maximum loss rate per block.

The goal of an hybrid scheme is to amortize the cost of a signature over several packets. Thus the greater the block size, the less often we will need to compute a signature. On the other hand, the block size influences the authentication delay and/or the sender buffer size, depending on which scheme of section 3.2 is chosen. As we said above, EC1 seems to be a good compromise in most situations with both a buffering and a maximum authentication delay of one block. Once a scheme is chosen, we recommend to chose the largest possible block size $b$ within the constraints of the application authentication delay requirements.

The parameter $p$ depends on the loss pattern of our network. In this work, we propose to refer to a model often suggested to describe bursty losses in Internet traffic which is a simple 2 state Markov chain [2, 18] also called the Gilbert model, where state 0 represents a packet received and state 1 a packet lost by the recipient. If $r$ denotes the probability of going from state 0 to state 1 and $q$ the probability of going from state 1 to state 0 we have the following transition matrix[5]: $M = \begin{bmatrix} (1-r) & r \\ q & (1-q) \end{bmatrix}$ The probability that $k$ consecutive packets are lost is equal to $(1-q)^{k-1}q$ which describes a distribution of mean $\mu = \frac{1}{q}$. The long term average loss rate $\pi_1$ is calculated as $\pi_1 = \frac{r}{r+q}$ (see also [2, 11]). We further note that Perrig *et al.* have used this model for their simulations in their own stream authentication scheme, EMSS[12].

## 4. Discussion

### 4.1. Computational Cost

For each block, the source needs to compute $b$ hash operations, a digital signature, and generate the 2 codes. Here, the hashing and signing costs are equivalent to other hybrid schemes as found in [12] or [4]. In the ideal case, where no loss occurs, the recipient just computes $b$ hashes and verifies a signature. If packets are lost some additional decoding operations are needed. The codes are used to recover hashes of packets, rather then the packets themselves, thus we will be manipulating small amounts of data. In traditional uses of Erasure Codes, the packet size $L$ is typically over a thousand bytes, while here we are looking at figures ranging from $L = 1$ to $L = 150$ bytes in the most extreme cases.

If we take a simple Reed-Solomon Erasure Code[14], the computational decoding cost is $\mathcal{O}(m.e.L)$ where $m$ is the number of original message packets, and $e$ the additional parities needed (corresponding to the loss) and $L$ the size of a packet. The coding cost is similarly in $\mathcal{O}(m.k.L)$ where $k$ is the number of parities. For demanding situations, we can turn to more efficient codes such as Tornado Codes[7], which achieve coding and decoding times in $\mathcal{O}((m+k).\ln(1/\varepsilon).L)$ (in [3] a value $\varepsilon \approx 0.05$ is suggested).

Compared to other hybrid live authentication streams, the main tradeoff of our scheme is in the additional computational cost generated by the erasure code. However, since we are operating on a small code packet size, the cost over a block should remain very reasonable. We will show in the next section that the substantial gain we can achieve in terms of overhead per packet is clearly worth the extra computational effort.

| $p\backslash b$ | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |
|---|---|---|---|---|---|---|---|
| 0.05 | 10 | 6 | 4 | 2 | 2 | 2 | 1 |
| 0.10 | 12 | 7 | 5 | 3 | 3 | 3 | 2 |
| 0.25 | 16 | 11 | 8 | 7 | 6 | 6 | 6 |
| 0.50 | 32 | 24 | 20 | 18 | 17 | 17 | 17 |
| 0.75 | 80 | 64 | 56 | 56 | 50 | 49 | 49 |

**Table 1. Overhead bytes per packets for different values of $p$ and $b$**

### 4.2. Overhead

**Evaluation** The overhead in bytes per packet of our 3 schemes is uniquely defined by the size of an authentication tag. Thus, recalling Proposition 2 in section 3 we can express the overhead as a function $\delta(b,p)$ of $p$, the maximum expected loss rate per block, and $b$, the number of packets in a block: $\delta(b,p) = \frac{\mathcal{R}_{\lfloor(1-p)b\rfloor}(s+\lceil p.b\rceil h)}{\lfloor(1-p)b\rfloor}$.

We would like to emphasize that this overhead *includes the signature overhead*. Table 1 presents a sampling of $\delta(p,b)$ for different values of $p$ and $b$, with $s = 128$ bytes (1024 bit RSA) and $h = 16$ (MD5[13]). Note that $\delta(b,p)$ remains surprisingly small if either $b$ large or $p$ is reasonably low.

**Case studies** To be more concrete we applied our scheme to the two case studies proposed by the authors of the EMSS[12] live stream authentication scheme. We recall their first case study:

A municipality wishes to collect traffic information from sensors distributed over the streets. The system requirements are as follows:

- The data rate of the stream is about 8 Kbps, about 20 packets of 64 bytes each are sent every second.
- The packet drop rate is at most 5% for some recipients, where the average length of burst drops is 5 packets.
- The verification delay should be less than 10 seconds.

We propose to use the ECU scheme since the sensors may have limited memory, thus the verification delay of 10 seconds allows us to use a block of 100 packets ($\frac{200}{2}$ since a block is authenticated by the next one).

Given the drop rate and the average length of bursts, we constructed a corresponding 2 state Markov chain and simulated it over 10000 blocks of 100 packets (for simulation techniques we referred to [6]). We found that 99% of those blocks experienced a loss less than 27 packets, thus we decided to chose $p = 0.27$. The overhead[1] per packet is then only $\delta(100, 0.27) = 8$ bytes !

[1]If we had chosen the EC1 scheme instead, we would have $b = 200$, $p = 0.2$ and $\delta(b,p) = 5$.

|  | $\pi_1$ | $\mu$ | $b$ | $p$ | $\delta(p,b)$ |
|---|---|---|---|---|---|
| Example 1 | 10% | 3 | 32 | 0.47 | 22 |
| Example 2 | 10% | 50 | 512 | 0.50 | 18 |
| Example 3 | 80% | 10 | 200 | 0.905 | 160 |
| Example 4 | 5% | 5 | 1024 | 0.1 | 2 |

**Table 2. A few case studies.**

The second case study proposed by Perrig *et al.* is related to real-time video broadcasting, with the following requirements:

- The data rate of the stream is about 2Mbps, or 512 packets of 512 bytes each every second.
- The packet drop rate is at most 60% for some recipients, with an average length of burst drops of 10 packets.
- The verification delay should be less than 1 second.

We propose the EC1 scheme and because of the verification delay, we have to limit $b$ to 512 packets. We simulated the corresponding Markov model over 10000 blocks and found that 99% of those blocks experienced a loss of less than 375 packets. We decided to chose $p = \frac{375}{512}$, which gives us an overhead per packet of $\delta(512, 0.73) = 45$ bytes.

As a complement to the two proposed scenarios above, Table 2 shows a few of our other simulation results, following the same approach as above for different average burst loss lengths $\mu$ and loss rates $\pi_1$. Example 1 shows that with a small block size, parameter $p$ is significantly higher than the network loss rate. Similarly, an extreme average burst length increases the value of $p$ as shown in example 2. Finally we have two extreme examples of the parameters of our scheme: first in a very lossy network which requires 160 bytes of overhead per packet which more than the size of a public key signature, and to finish we have an ideal case, with a small loss and a long block size which gives us a surprisingly low overhead per packet of 2 bytes !

## 5. Comparison

### 5.1. Hash Trees

The authors of the Hash Trees construction[17] use Merkle[8] authentication techniques to construct a balanced binary hash tree over a block of $b$ packets. To allow the received packets to be authenticated independently (and make the scheme joinable on any packet), each packets contains a set of $(ln_2(b) - 1)$ hashes needed to recover the root of the hash tree as well as a digital signature of the root. This leads

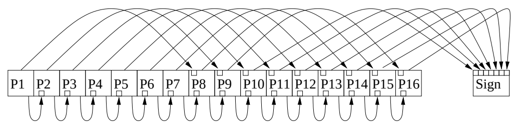

**Figure 1. Hash chain resisting to bursts of 6 packets in a 16 packet block.**

to an overhead per packet of $s + (ln_2(b) - 1).h$ bytes which is even larger than than the "sign each" approach, though it only requires one signature computation per block. Just like EC1 (and EC2), the scheme requires the sender to buffer the whole block before the first packet of that block can be sent.

## 5.2. Hash Chains

Golle and Modadugu[4] proposed a stream authentication mechanisms designed to tolerate the loss of packets in bursts of at most $\beta$ packets in a block. They construct a directed acyclic graph between the packets of the block, by putting the cryptographic hash of a packet in one or several other packets. If a packet $P$ is signed then any packets $P'$ for which there exists a path in the graph joining $P'$ to $P$ can be authenticated. The authors of [4] propose methods to design such acyclic graphs in an optimal way regarding bursty packet losses. Their simplest scheme is constructed as shown on the example of figure 1: the hash of a packet $P_i$ is stored both as part of the following packet $P_{i+1}$ and as part of $P_{i+1+\beta}$. Finally the hashes of the last $(\beta + 1)$ packets are sent, along with a signature of these $(\beta + 1)$ hashes for verification.

The same authors further refined their hash chain construction, to create "Augmented Chains", which require to buffer a few packets, but allows a smaller set of hashes to be signed at the end. The principle remains the same and we refer the reader to [4] for details. Their first scheme can tolerate several bursts in a block while the "augmented chain" construction may have difficulties if there are several bursts in the same block, consequently we will focus on their first scheme in this comparison.

**Hash Chain Overhead:** The authors of [4] do not detail how to choose $\beta$ nor do they provide a clear method to deal with signature loss except to suggest the transmission of several copies of the signature. If these signatures are transmitted far enough apart, we can consider that their loss probabilities are uncorrelated. If we assume that $\gamma$ signatures are transmitted, we can approximate the overhead as $\delta_{HC}(\gamma, b) = \gamma\frac{(\beta+1).h+s}{b} + 2h$ bytes per packets, with the notations already used throughout this work. The size of $b$ is essentially constrained by the authentication delay, which here is at most the distance between the first packet of the block and the $\gamma^{th}$ redundant signature that is transmitted for that block.

Recalling the Markov chain model of section 3 we know that the probability that a burst of $k$ lost packets occurs is $q.(q-1)^{(k-1)}$ with an average length of $\frac{1}{q}$ packets in a burst. Consequently we will choose $\beta$ in the hash chain such that the probability that a burst exceeds $\beta$ is low, for example such that $1 - \sum_{k=(\beta+1)}^{\infty} q(1-q)^{k-1} \geq 99\%$. If we refer to the two case studies we borrowed from EMSS in section 3, we would have:

**Case 1:** We propose $b = 160$, $\gamma = 2$, $\beta = 21$ since $q = 0.2$. We would transmit the first signature at the end of the block and the second signature 20 packets later (1 second). The probability that one of the signature arrives is approximately $1 - 0.05^2 = 0.9975$ and the overhead per packet is $\delta_{HC}(\gamma, b) \approx 38$ bytes.

**Case 2:** This case is more problematic because a individual signature has a good probability of being lost. Indeed, if we take $\gamma = 8$ the probability that one redundant signature at least arrives is $1-0.6^8 \approx 0.99$ (if we take $\gamma = 4$ the probability is lowered to $0.87$). But this means that each block is transmitted along with 4 to 8 signatures and it becomes difficult to define a reasonable size for $b < 512$. If we chose $b$ small then we need to compute several signatures per second and we need to send several copies of each of them during the same time (without a guaranty that losses will be independent). If we chose $b$ larger then the probability of authenticating a packet within the authentication delay becomes lower. As a indication, if $b = 256$, $\gamma = 8$, $\beta = 43$, we have $\delta_{HC}(\gamma, b) \approx 58$ bytes.

No matter how good the network conditions are and no matter how long the block size is, the hash chains have at least an overhead of $2.h$ bytes per packets (with a few extra bytes for the signature). Comparatively, our scheme has clearly a lower overhead when the network is not too lossy, with such extremes shown as in Example 4 in table 2. For more lossy streams, our scheme maintains a high authentication probability despite the losses, without encountering the problems we described above in Case 2.

## 5.3. EMSS

Perrig et al. used a similar hash chain idea in their EMSS[12] scheme, with an more concrete method to deal with block signature loss. As opposed to [4] which uses a deterministic edge relationship pattern among the packets in the chain, EMSS uses randomly distributed edges. Moreover, packets are chained across blocks, thus event if all the redundant signatures pertaining to a block are lost, the signature in the next block can be used to authenticate the data (potentially out of authentication delay). They performed several simulations in order to tune the right number of hashes to include in each packet depending on the loss characteristics of the stream. The signature of a block is transmitted several times to allow it to reach the recipient with high probability, depending on the characteristic of the network.

Since we borrowed our 2 test cases directly from EMSS, we can recall their results here as a comparison. The simulations conducted in EMSS give an overhead of $22$ bytes in *Case 1* (with an average verification probability per packet of 98,7%) and an overhead of $55$ bytes in *Case 2* (with a minimum verification probability of 90%). In *Case 2*, the signature of a block alone which is transmitted twice has only an estimated probability of arrival of $0.64 \approx 1 - 0.6^2$, but since there is linking between blocks a packet may be verified by the signature of future blocks, however in this case we understand that the verification delay of a packet will be exceeded. We would also like to highlight that their scheme used 80 bit hashes while we use 128 bit hashes (MD5). A similar value in our scheme would have given an even lower overhead per packet and also a lower overhead in the Hash Chain construction.

Despite longer hashes, in both cases, our scheme has lower overhead and a higher probability of block verification within the required authentication delay.

### 5.4. TESLA

Perrig *et al.*[12] proposed a very efficient time based stream authentication scheme called TESLA. It provides source authentication but does offer nonrepudiation. It tolerates arbitrary packet loss with a reasonably low overhead. Its main drawback is that it requires a secure clock synchronization between the source and the recipients which may not be always feasible in a large multicast group. Moreover, all secure clock servers become potential targets for adversaries who wish to defeat the authentication scheme. The scheme relies on the reliable transmission of a signature as a commitment during initialization, thus it is worth noting that without the regular transmission of additional commitments, the TESLA scheme is only joinable at the beginning of the stream. For these reasons, our scheme may be a more practical alternative to TESLA in some scenarios.

## 6. Conclusion

In this work we propose a new approach to live lossy stream authentication, which is joinable on block boundaries. Where previous proposals used hash linking, we use erasure codes to achieve a lower overhead per packet. Moreover, we propose a concrete mechanism describing how to transmit the authentication information as well as the signature associated to a block with equivalent recovery probabilities. We proposed buffered and unbuffered variations of our scheme which offer an interesting alternative to other live stream authentication mechanism in many situations.

## References

[1] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *proceedings of EuroCrypt '94*, volume 950 of *LNCS*. Springer-Verlag, 1995.

[2] J.-C. Bolot, S. Fosse-Parisis, and D. F. Towsley. Adaptive FEC-based error control for internet telephony. In *INFOCOM '99*, pages 1453–1460, 1999.

[3] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *proceedings of ACM SIGCOMM '98*, September 1998.

[4] P. Golle and N. Modadugo. Streamed authentication in the presence of random packet loss. In *to appear in NDSS 2001.*, 2001.

[5] C. M. Grinstead and J. L. Snell. *Introduction to Probability*. McGraw Hill, 2000.

[6] O. Häggström. *Finite Markov Chains and Algorithmic Applications*. Cambridge, June 2002.

[7] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann. Practical loss-resilient codes. In *ACM Symposium on Theory of Computing*, pages 150–159, 1997.

[8] R. C. Merkle. A certified digital signature. In *CRYPTO '89*, pages 218–238. Springer-Verlag, August 1989.

[9] S. Miner and J. Staddon. Graph-based authentication of digital streams. In *2001 IEEE Symposium on Security and Privacy*, May 2001.

[10] National Institute of Standards and Technology. Secure hash standard, 1995.

[11] A. Pannetrat and R. Molva. Real time multicast packet authentication. Technical report, Institut Eurécom, March 2002.

[12] A. Perrig, R. Canetti, J. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, May 2000.

[13] R. Rivest. The MD5 message-digest algorithm. In *Request for Comments*, volume (RFC) 1321, April 1992.

[14] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACMCCR: Computer Communication Review*, 27, 1997.

[15] P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *Proceedings of the 6th ACM conference on Computer and Communications Security*, pages 93–100, 1999.

[16] RSA Security Inc. PKCS-1 v2.1: RSA cryptography standard, 1999.

[17] C. K. Wong and S. S. Lam. Digital signatures for flows and multicasts. *IEEE/ACM Transactions on Networking*, 7(4):502–513, 1999.

[18] M. Yajnik, S. Moon, J. Kurose, and D. Towsley. Measurement and modeling of the temporal dependence in packet loss. In *IEEE INFOCOM*, New York, Mar. 1999.