

A Coding Approach to the Multicast Stream Authentication Problem*

Christophe Tartary^{1,2}, Huaxiong Wang^{1,3} and Josef Pieprzyk³

¹Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University
Singapore

²Institute for Theoretical Computer Science
Tsinghua University
Beijing, 100084
People's Republic of China

³Centre for Advanced Computing, Algorithms and Cryptography
Department of Computing
Macquarie University
NSW 2109 Australia

{ctartary, HXWang}@ntu.edu.sg
josef@ics.mq.edu.au

Abstract

We study the multicast stream authentication problem when an opponent can drop, reorder and introduce data packets into the communication channel. In such a model, packet overhead and computing efficiency are two parameters to be taken into account when designing a multicast stream protocol. In this paper, we propose to use two families of erasure codes to deal with this problem, namely, rateless codes and maximum distance separable codes. Our constructions will have the following advantages. First, our packet overhead will be small. Second, the number of signature verifications to be performed at the receiver is $O(1)$. Third, every receiver will be able to recover all the original data packets emitted by the sender despite losses and injection occurred during the transmission of information.

Keywords: Stream Authentication, Polynomial Reconstruction, Rateless Codes, Erasure Codes.

1 Introduction

Multicast protocols enable data to be transmitted from one sender to many receivers via a communication network such as the Internet. The applications are as various as pay-TV, online games and military defense systems for instance. Nevertheless, large-scale broadcasts prevent lost content from being retransmitted since the loss of any piece of data could generate an overwhelming number of redistribution requests at the sender. In addition, the network can be under the influence of malicious users performing harmful actions on the data stream. Thus, the security of broadcast transmission schemes depends on both network properties and opponents' computational power. In this paper, we will consider that opponents have bounded computational abilities.

Some applications for television or stock market will diffuse a long stream of data whose content is to be authenticated by the receivers within a short period of time upon reception. Since many multicast protocols transfer private or sensitive information, non-repudiation of the sender is required for most of them. In addition, it should also be pointed out that most channels used for multicast only provide a best effort delivery of data like the Internet with the User Datagram Protocol.

Two important concerns of the multicast authentication problem are the network bandwidth availability and the receivers' computational abilities. Indeed, large packets may create irregular throughput of data which sometimes results in congestion of the network information flow. On the other hand, receivers with limited computational abilities will require more time to authenticate data delaying the stream play. Therefore, a stream authentication protocol is to minimize both

*This paper is the extended version of the articles [65, 64] appearing in the proceedings of IWSEC 2006 and CANS 2006. Its original version appears in the International Journal of Information Security, vol. 7, no. 4, pp 265 - 283, August 2008, Springer - Verlag.

packet¹ overhead and computational cost.

In recent years, several protocols were designed to deal with the multicast authentication problem [11]. The simplest technique to ensure non-repudiation of information is to sign each packet using a digital signature (*sign-each approach*). Unfortunately, this solution is impractical as digital signatures are time expensive to generate and verify while k -time signatures exhibit a large length [19, 57]. That is why a common approach is to generate a single signature and to amortize its communication and computation overheads over several packets using hash functions for instance.

By appending the hash of each packet to several followers according to some specific patterns, Perrig *et al.* [50, 51], Golle and Modadugu [20] and Miner and Staddon [40] designed schemes dealing with packet loss. One signature was generated from time to time and was always assumed to be received. In these contributions, authors modeled the network packet loss by a k -state Markov chain [17, 49, 69] and provided bounds on the packet authentication probability. Gao and Yao [18] proposed to use online/offline signature to speed up signing and verifying time for these schemes. Unfortunately, all these protocols rely on reception of signed packets.

To overcome this problem, one solution is to split the signature into k smaller parts where only ℓ of them ($\ell < k$) are enough for recovery. Along this line, several schemes were developed [2, 45, 46, 47, 48] but none of them tolerates a single packet injection. In 2003, Lysyanskaya *et al.* [35] designed a technique (called in this paper LTT) resistant to packet loss and data injections using Reed-Solomon codes [55] where the number of signature verifications to be performed per block² turns out to be $O(1)$ as a function of the block length n . In 2004, Karlof *et al.* developed a protocol called PRABS [27] using a Maximum Distance Separable (MDS) code along with a one-way accumulator [5, 7, 42, 43] based on a Merkle hash tree [39] requiring less signature verifications than LTT. Unfortunately, PRABS's augmented packets³ must carry $\lceil \log_2 n \rceil$ hashes which is much larger than for the constructions we propose in this paper. Note that the elliptic curve-based accumulator developed by Nguyen [42] overcomes this drawback. Like PRABS, each received element has to be sorted according to its accumulated value. For Nguyen's construction, this requires to evaluate two pairings which is the cost of verifying one signature produced by Boneh *et al.*'s digital signature scheme [8]. Nevertheless, Boneh *et al.* noticed that this operation was much slower than a RSA signature verification which was reinforced by Barreto *et al.* [6]. So, using Nguyen's accumulator is not suitable in our context since even the prohibitive sign-each approach would be more efficient.

Our approach is motivated by the following observation. A necessary condition for all these schemes to authenticate a packet P (at the receiver) is to get an element \tilde{P} containing P along with some hashes [20, 40, 51, 67] or code symbols [35, 45, 63]. If \tilde{P} is dropped then P is definitely lost since such a \tilde{P} is unique for each scheme. We propose to use erasure correcting codes to overcome this problem. As the previous techniques, we will process the data stream per block of n packets: P_1, \dots, P_n . The two constructions we propose can be seen as extensions of LTT and PRABS which enable any receiver to recover all data packets P_1, \dots, P_n despite loss incurred during transmission. This constitutes a major improvement from existing approaches in the way that receivers not only authenticate what they received but also reconstruct what was lost. This is particularly beneficial when P_1, \dots, P_n represent audio or video information where our techniques prevent frozen images and audio gaps to happen.

In both constructions, a digital signature will be used to ensure non-repudiation and to enable new members to join the communication group at any block boundary.

Our first scheme uses **Luby Transform (LT)** codes to encode blocks of n data packets P_1, \dots, P_n into \mathcal{N} symbols $E_1, \dots, E_{\mathcal{N}}$ (the value of \mathcal{N} is specified in Section 4). LT codes were introduced by Luby [32] as the first practical realization of rateless codes to illustrate the Digital Fountain concept [9]. These codes are constructed in such a way that there exists a threshold value m (depending on n) such that any subset of $\{E_1, \dots, E_{\mathcal{N}}\}$ having at least m distinct elements can be used to recover all n original packets P_1, \dots, P_n with good probability. By representing $E_1, \dots, E_{\mathcal{N}}$ as coefficients of a particular polynomial and carefully choosing \mathcal{N} , the receiver will be able to run a reconstruction algorithm due to Guruswami and Sudan [24] and will recover that polynomial despite potential data injections performed by malicious users.

Since we will use the same opponent model as Lysyanskaya *et al.*, we will prove that our scheme is as secure as LTT and exhibits a minimal lower bound on the packet authentication probability which can be chosen arbitrary close to 1. Since the security of our construction depends on the consistency of the LT decoding (while LLT relies on Reed-Solomon codes' one), we will compare LT codes to other families of rateless codes including Online and Raptor codes [37, 60]. We will show that it is possible to achieve reasonable packet overhead by using a modified version of LT codes. We will also enlighten that Raptor codes can provide good practical implementations for our scheme if they are used instead of LT codes. The reader may be aware that Raptor codes have recently been used in many applications related to the distribution

¹Since the stream size is large, it is divided into small fixed-size entities called *packets*.

²In order to be processed, packets are gathered into fixed-size sets called *blocks*.

³We call *augmented packets* the elements sent into the network. They generally consist of the original data packets with some redundancy used to prove the authenticity of the element.

of digital content [1, 68].

As we will see, rateless codes only requires packet XOR-ings to encode and decode data which is suitable for devices with limited computational power. Nevertheless, when receivers have hardware allowing more complex arithmetic operations to be performed (such as field operations for instance), one may think about using an approach similar to PRABS. Our second protocol will allow complete recovery of the whole data stream (with probability 1) using a MDS codes construction developed by Lacan and Fimes in 2004. Based on their work [30], we will argue that applying their code construction results in a better encoding/decoding complexity than using most other MDS codes. When compared to Reed-Solomon codes (as used for LTT) our technique will generate a slight increase of computing complexity at the sender (which can be compensated by his larger computational power) whereas the complexity at the receivers will be reduced. We will also see that our packet overhead is smaller than PRABS for practical applications.

Contrary to [35] where only an asymptotic study of LTT was performed, we will derive an upper bound on the number of signature verifications to be executed per block for both schemes. This bound will be valid for any block length and will turn out to be $O(1)$ as for LTT and PRABS. Such a bound is valuable for practical applications since the block length is always finite. It allows receivers to get an upper bound on the time spent to verify signatures and therefore on the delay⁴ between reception of information and authentication of correct packets.

This paper is organized as follows. In the next section, we present our network model as well as a few results from [30, 32]. In Section 3, we describe our general approach to the stream authentication problem. Our constructions are presented in Section 4 and Sect. 5 where we demonstrate their security and study their efficiency. In Section 6, we compare the benefits and drawbacks of our two schemes. The last section will summarize our contributions to the multicast authentication problem.

2 Preliminaries

We now introduce the terminology and assumptions we will use in this paper. First, we need to define our network model. Then, we will describe the two code constructions by Luby and Lacan and Fimes. Finally, we will recall an algorithm by Guruswami and Sudan achieving polynomial reconstruction. As in [35, 63], it will be used to deal with packet injections.

2.1 Network Model

We stated in the introduction of this paper that the elements sent into the network consists of the original data packets with some redundancy used to provide authenticity and are called augmented packets. We consider that the communication channel is under control of an opponent \mathcal{O} who can drop and rearrange packets of his choice. He is also allowed to inject bogus data into the network. Since our primary concern is the multicast authentication problem, we can assume that a reasonable number of original augmented packets reaches the receivers and not too many incorrect elements are injected by \mathcal{O} . Indeed, if too many original packets are dropped then data transmission becomes the main domain of investigation since the small number of received elements would be probably useless even authenticated. On this other hand, if \mathcal{O} injects a large number of forged packets then the main problem to be solved becomes increasing the resistance against denial-of-service attacks. In order to build our signature amortization schemes, we need to split the data stream into blocks of n packets: P_1, \dots, P_n .

We define two parameters: α ($0 < \alpha \leq 1$) (the *survival* rate) and β ($\beta \geq 1$) (the *flood* rate). It is assumed that at least a fraction α and no more than a multiple β of the number of augmented packets are received. This means that, when \mathcal{A} augmented packets are sent into the network, at least $\lceil \alpha \mathcal{A} \rceil$ of them are received amongst a total which does not exceed $\lfloor \beta \mathcal{A} \rfloor$ elements. Notice that we will have $\mathcal{A} = n$ for our MDS code-based scheme while $\mathcal{A} > n$ using LT codes.

We also draw the reader's attention to the fact that we are not interested in the cases ($\alpha = 1$) and ($\beta = 1$). Indeed, in the first case, all original data packets are received. Thus, we only need to distinguish correct elements from bogus ones. This can be done using Wong and Lam's technique [67]. In the second case, there are no packet injections from \mathcal{O} . Thus, using an erasure code (as in [13] for instance) is enough to recover all augmented packets and so P_1, \dots, P_n . Therefore, in this work, we will only study the case: $0 < \alpha < 1 < \beta$.

2.2 Code Construction

2.2.1 LT Codes

We briefly describe how to generate outputs for LT codes and how to decode data. A complete description of both processes can be found in [32].

⁴This delay is called *authentication delay* of the scheme.

Encoding. We have a fixed number of input symbols denoted by I_1, \dots, I_K . In order to generate a new encoding symbol E , we use a probabilistic distribution called the Robust Soliton distribution to choose the degree⁵ d of the symbol E . We randomly pick d elements amongst the input symbols: I_{i_1}, \dots, I_{i_d} ⁶. We generate E as the XOR of I_{i_1}, \dots, I_{i_d} . Using this process, we can generate as many encoding symbols as we want since we only need to run the Robust Soliton distribution to get a new one.

Decoding. When the receiver gets N encoding symbols E_1, \dots, E_N , he first builds the bipartite graph used to compute E_1, \dots, E_N ⁷. We would like to point out that it can happen that not every I_i is on the left hand side. This is true in particular if N is small and the encoding symbols have small degrees. At the beginning of the decoding process, no I_j 's have been covered⁸. They are initialized with 0's. We first release⁹ all E_ℓ 's with a single adjacent vertex to cover their unique neighbor. The set of covered input symbols not yet processed is called the *ripple* and denoted R . All previous covered symbols belong to R . At each step, one element I_j is processed as follows:

1. Each neighbor N_j^ℓ of I_j has its value XOR-ed with I_j 's.
2. I_j is removed as a neighbor of these elements N_j^ℓ . That is, the corresponding edges are removed from the graph.
3. For each N_j^ℓ having one remaining neighbor in the new graph, N_j^ℓ is released from the graph and covers its remaining neighbors which are added to R (for those which were not already in).
4. I_j is released from R (because it has no neighbors any longer).

Steps 3 and 4 make the size of R vary. The decoding process ends when R is empty. It is successful when I_1, \dots, I_K have been released from R . We will use the following theorem to deal with packet loss occurring during data transfer.

Theorem 1 ([32]) For $\delta \in (0, 1)$, the decoding process fails with probability at most δ from any set of $N := K + (R + \frac{R}{2} + \dots + \frac{R}{K-R}) \ln(\frac{R}{\delta})$ encoding symbols where $R := c \ln(\frac{K}{\delta}) \sqrt{K}$ for a positive constant c determined within the Robust Soliton distribution.

In [10], it is proved that c has to be chosen as:

$$\frac{1}{K-1} \cdot \frac{\sqrt{K}}{\ln(K/\delta)} \leq c \leq \frac{1}{2} \cdot \frac{\sqrt{K}}{\ln(K/\delta)}$$

In this paper we consider

$$\frac{1}{K-1} \cdot \frac{\sqrt{K}}{\ln(K/\delta)} \leq c \leq \frac{1}{2} \cdot \frac{K^\eta}{\ln(K/\delta)}$$

for some constant (independent from K) value η in $(0, \frac{1}{2})$ so that our packet overhead remains reasonable (see Appendix A).

2.2.2 MDS Codes

In this part, we focus on linear codes. A linear code of length N , dimension K and minimum distance D is denoted $[N, K, D]$. The Singleton bound states that any $[N, K, D]$ code satisfies: $D - 1 \leq N - K$ [36]. It is known that any $[N, K, D]$ code can correct up to $D - 1$ erasures [70]. Thus, a $[N, K, D]$ code cannot correct more than $N - K$ erasures. In order to maximize the efficiency of our construction, we are interested in codes correcting exactly $N - K$ erasures. These codes are called *Maximum Distance Separable (MDS)* codes [36].

Now, we describe the MDS code construction developed in [30]. We will work over the field \mathbb{F}_{2^q} . Every element of \mathbb{F}_{2^q} can be represented as a polynomial of degree at most $q - 1$ over \mathbb{F}_2 [31]. Operations in \mathbb{F}_{2^q} are performed modulo a polynomial $Q(X)$ of degree q which is irreducible over \mathbb{F}_2 . From [30], we have:

Theorem 2 Let $V(a_1, \dots, a_K)$ be a non-singular $K \times K$ Vandermonde matrix and let $V(b_1, \dots, b_{N-K})$ be a $K \times (N - K)$ matrix $(b_i^{j-1})_{i=1, \dots, N-K}^{j=1, \dots, K}$ (with convention $0^0 = 1$). Consider the $K \times K$ identity matrix I_K . Then, the linear code defined by the generator matrix:

$$G := [I_K | V(a_1, \dots, a_K)^{-1} V(b_1, \dots, b_{N-K})]$$

is MDS if and only if the a_i, b_j are N pairwise distinct elements.

⁵Any LT code can be represented as a bipartite graph with I_1, \dots, I_K as the left hand side vertices and all encoding symbols as right hand side vertices. An edge is drawn between I_j and the encoding symbol E if I_j has been used to compute E . I_j is said to be a *neighbor* of E (and vice versa). We use the term *degree* to denote the number of neighbors a symbol has.

⁶This is how we build the bipartite graph representing the LT code.

⁷The positions of the input symbols XOR-ed to build an encoding symbol E_i are sent along with E_i [25].

⁸An input symbol I_j is said to be *covered* when it is the only adjacent vertex of an encoding symbol E_ℓ . The covering operation is a XOR of the current value of I_j with E_ℓ .

⁹A symbol is said to be *released* when we remove its representing vertex from the graph.

Notice that due to the presence of the identity matrix I_k , each message to be encoded will appear as a part of its corresponding codeword. This means that this MDS code is in a *systematic* form. We now introduce the algorithms EncodeMDS and DecodeMDS which will be used as subroutines in our work. We will encode K symbols S_1, \dots, S_K into N modified symbols $\hat{S}_1, \dots, \hat{S}_N$. Each S_i and \hat{S}_j belongs to $(\mathbb{F}_{2^q})^r$. The choice of the efficiency parameter r will be explained in Section 5.

Algorithm 1 EncodeMDS

Input: The code length and dimension N and K , the polynomial $Q(X)$ of degree q , the generating matrix G , K symbols S_1, \dots, S_K and the efficiency parameter r .

1. Parse each symbol S_i into r field elements as: $S_i := S_i^1 \parallel \dots \parallel S_i^r$. Build r messages as: $\forall j \in \{1, \dots, r\} m_j := (S_1^j \dots S_K^j)$.
2. Encode the r messages into r codewords as: $\forall j \in \{1, \dots, r\} c_j := m_j G$.
3. Write each codeword as: $c_j := (c_1^j \dots c_N^j)$ and build the N modified symbols as: $\forall j \in \{1, \dots, N\} \hat{S}_j := c_1^j \parallel \dots \parallel c_r^j$.

Output: N modified symbols: $\hat{S}_1, \dots, \hat{S}_N$.

Algorithm 2 DecodeMDS

Input: The code length and dimension N and K , the polynomial $Q(X)$ of degree q , the generating matrix G , $T(\geq K)$ elements $\{(j_i, \hat{S}_{j_i}), 1 \leq i \leq T\}$ and the efficiency parameter r .

1. Reorder the T elements to have $j_1 < \dots < j_T$ and pick the first K elements. Parse the \hat{S}_{j_i} 's as: $\forall i \in \{1, \dots, K\} \hat{S}_{j_i} = c_1^{j_i} \parallel \dots \parallel c_r^{j_i}$ and write: $\forall i \in \{1, \dots, r\} c_i' := (c_i^{j_1} \dots c_i^{j_K})$.
2. Build G' as the restriction of G to columns j_1, \dots, j_K and compute r messages as: $\forall i \in \{1, \dots, r\} m_i := c_i' G'^{-1}$.
3. Write each message as: $m_j := (m_1^j \dots m_i^K)$ where each m_i^j is $\deg(Q(X))$ bits long. Recover the K symbols as: $\forall i \in \{1, \dots, K\} S_i = m_1^i \parallel \dots \parallel m_r^i$.

Output: K symbols: S_1, \dots, S_K .

Notice that the polynomial $Q(X)$ is used at Step 2 of EncodeMDS and DecodeMDS when performing field operations.

2.3 Polynomial Reconstruction Problem

In [24], Guruswami and Sudan developed an algorithm to solve the polynomial reconstruction problem. They proved that if T points were given as input then their algorithm [Poly-Reconstruct](#) output the list of all polynomials of degree at most K passing through at least N of the T points provided: $T > \sqrt{KN}$. We will use a modified version of Poly-Reconstruct that we call [MPR](#). Denote \mathbb{F}_{2^q} the field representing the coefficients of the polynomial. As before, we denote $Q(X)$ the polynomial used to perform operations in that field.

Algorithm 3 MPR

Input: The maximal degree K of the polynomial, the minimal number N of agreeable points, T points $\{(x_i, y_i), 1 \leq i \leq T\}$ and the polynomial $Q(X)$ of degree q .

1. If there are no more than \sqrt{KN} distinct points then the algorithm stops.
2. Using $Q(X)$, run Poly-Reconstruct on the T points to get the list of all polynomials of degree at most K over \mathbb{F}_{2^q} passing through at least N of the points.
3. Given the list $\{L_1(X), \dots, L_\mu(X)\}$ obtained at Step 2. For each polynomial $L_i(X) := \mathcal{L}_{i,0} + \dots + \mathcal{L}_{i,K} X^K$ where $\forall i \in \{0, \dots, K\} \mathcal{L}_{i,j} \in \mathbb{F}_{2^q}$, form the elements: $\mathcal{L}_i := \mathcal{L}_{i,0} \parallel \dots \parallel \mathcal{L}_{i,K}$.

Output: $\{\mathcal{L}_1, \dots, \mathcal{L}_\mu\}$: list of candidates

3 Overview of our Approach

In this section, we give a general overview of our two protocols. As in [27, 35, 63], we need a collision resistant hash function h [52] and an unforgeable digital signature scheme $(\text{Sign}_{\text{SK}}, \text{Verify}_{\text{PK}})$ [61] the key pair of which (SK, PK) is

generated by an algorithm KeyGen.

LT Code-based Construction. From the n data packets P_1, \dots, P_n , we want to generate $\mathcal{A} = \mathcal{N}$ augmented packets $AP_1, \dots, AP_{\mathcal{N}}$ such that if at most $\mathcal{N} - \lceil \alpha \mathcal{N} \rceil$ of them are lost during transmission then the receiver can still recover all the P_i 's with probability at least $1 - \delta$ (with δ close to 0). Thus, to deal with erasures, we need to generate at least $\mathcal{N} := \lceil \frac{m}{\alpha} \rceil$ symbols $E_1, \dots, E_{\mathcal{N}}$ from P_1, \dots, P_n using the LT code where $m := n + (R + \frac{R}{2} + \dots + \frac{R}{n-R}) \ln(\frac{R}{\delta})$ and $R := c \ln(\frac{n}{\delta}) \sqrt{n}$ (see Theorem 1). In order to provide non-repudiation and to deal with bogus injections, we hash the symbols along with the positions of their neighbors (necessary to rebuild the graph when decoding) and sign the concatenation $h_1 \parallel \dots \parallel h_{\mathcal{N}}$. As in [35], we build a polynomial $A(X)$ of degree at most $\rho \mathcal{N}$ (for some rational constant ρ), the coefficients of which represent $h_1 \parallel \dots \parallel h_{\mathcal{N}} \parallel \sigma$ (σ is the signature). We build the \mathcal{N} augmented packets as: $\forall i \in \{1, \dots, \mathcal{N}\} AP_i := \text{BID} \parallel i \parallel E_i \parallel N_i^1 \parallel \dots \parallel N_i^{d_i} \parallel A(i)$ where BID represents the position of the block P_1, \dots, P_n within the whole data stream and d_i is the degree of E_i .

Upon reception of data, the receiver checks the signature by reconstructing the polynomial $A(X)$ using MPR. Once the signature σ is verified, the receiver knows the original hashes $h_1, \dots, h_{\mathcal{N}}$. Thus, he can identify the correct E_i 's and their corresponding neighbors' positions $N_i^1, \dots, N_i^{d_i}$ from the list of elements he has got. According to the definition of α and \mathcal{N} , there must be at least m correct symbols from $E_1, \dots, E_{\mathcal{N}}$ in his list. Finally, he corrects the erasures using the LT code and recovers the n data packets P_1, \dots, P_n with probability at least $1 - \delta$.

MDS Code-based Construction. From the n data packets P_1, \dots, P_n , we will construct $\mathcal{A} = n$ augmented packets AP_1, \dots, AP_n such that if at most $n - \lceil \alpha n \rceil$ of them are lost during transmission then the receiver can still recover all the P_i 's. Thus, we need to encode these n packets using a $[n, \lceil \alpha n \rceil, n - \lceil \alpha n \rceil + 1]$ code. To perform this encoding, the size of elements forming the code's alphabet will be larger than the size of a data packet. In order to provide non-repudiation and to deal with bogus injections, we hash the modified symbols $\hat{S}_1, \dots, \hat{S}_n$ (generated by the MDS code) and sign the concatenation $h_1 \parallel \dots \parallel h_n$. As before, we build a polynomial $A(X)$ of degree at most ρn , the coefficients of which represent $h_1 \parallel \dots \parallel h_n \parallel \sigma$ (σ is the signature). We build the augmented packets as: $\forall i \in \{1, \dots, n\} AP_i := \text{BID} \parallel i \parallel \hat{S}_i \parallel A(i)$.

As previously, upon reception of data, the receiver checks the signature by reconstructing the polynomial $A(X)$ using MPR. Once the signature σ is verified, the receiver knows the original hashes h_1, \dots, h_n . Thus, he can identify the correct \hat{S}_i 's amongst the list of elements he has got. According to the definition of α , there must be at least $\lceil \alpha n \rceil$ symbols from $\hat{S}_1, \dots, \hat{S}_n$ in his list. Finally, he corrects the erasures using the MDS code and recovers the n data packets P_1, \dots, P_n .

4 LT Codes for Multicast Stream Authentication

In this section, we will describe a multicast authentication protocol using LT codes which is robust against packet loss and data injection. Our technique also allows any new user to join the communication group at any block boundary. We will demonstrate its security, exhibit a minimal bound for the packet recovery probability and show that the number of signature verifications per block at the receiver is $O(1)$ as a function of the block length n . Finally, we compare LT codes to other families of rateless codes to be used in our context.

4.1 Our Authentication Protocol

The data stream is processed per block of n packets: P_1, \dots, P_n . For our construction, we assume that α, β and δ are rational numbers. Thus, we can represent them over a finite number of bits using their numerator and denominator. In order to run Poly-Reconstruct as a subroutine of MPR, we have to choose $\rho \in (0, \frac{\alpha^2}{\beta})$. Notice that ρ has to be rational since $\rho \mathcal{N}$ is an integer. Since c is a fixed value of the Robust Soliton distribution, \mathcal{N} only depends on n, δ and α . Therefore, without loss of generality, one can consider that the value ρ is uniquely determined when n, α, β and δ are known. Table 1 summarizes the scheme parameters which are assumed to be publicly known.

n : Block length	δ : Decoding failure probability
α : Survival rate	β : Flood rate
A list of irreducible polynomials over \mathbb{F}_2	

Table 1: Public parameters for our LT code-based scheme.

The hash function h as well as Verify and PK are also assumed to be publicly known. We did not include them in Table 1 since they can be considered as general parameters. For instance, h can be SHA-256 [41] while the digital signature can be a 1024-bit RSA signature [56]. We denote \mathcal{H} the digest bit length and \mathcal{S} the bit length of a signature. Since h and the digital signature are publicly known, so are \mathcal{H} and \mathcal{S} . We denote τ_{par} the tag representing the communication parameters, namely: $\tau_{\text{par}} := n \parallel \alpha \parallel \beta \parallel \delta$. It is assumed that the list of polynomials contains a single polynomial $\mathcal{Q}(X)$ per degree value $\deg(\mathcal{Q}(X))$. We now present the algorithm used by the sender.

Algorithm 4 Authenticator1

Input: The secret key SK, the block number BID, Table 1 and n data packets P_1, \dots, P_n .

1. Compute $\mathcal{N} := \lceil \frac{m}{\alpha} \rceil$ where m is defined in Section 3. Consider the n packets as input symbols for the LT code and build \mathcal{N} encoding symbols: $E_1, \dots, E_{\mathcal{N}}$. Each symbol E_i is associated with the positions of its d_i neighbors $N_i^1, \dots, N_i^{d_i}$. Compute the hashes: $\forall i \in \{1, \dots, \mathcal{N}\} h_i := h(E_i \| N_i^1 \| \dots \| N_i^{d_i})$.
2. Compute the block signature σ as: $\sigma = \text{Sign}_{\text{SK}}(h(\text{BID} \| \tau_{\text{par}} \| h_1 \| \dots \| h_{\mathcal{N}}))$ and form the authentication tag $\tau = h_1 \| \dots \| h_{\mathcal{N}} \| \sigma$.
3. Denote ξ the smallest element of \mathbb{N} such that:

$$\left\lceil \frac{\mathcal{H}\mathcal{N} + \mathcal{S} + \xi}{\rho\mathcal{N} + 1} \right\rceil \geq \lceil \log_2 \mathcal{N} \rceil \quad (1)$$

Denote q the left hand side of Inequality (1). Write τ as the concatenation $a_0 \| \dots \| a_{\rho\mathcal{N}}$ of $(\rho\mathcal{N} + 1)$ elements of \mathbb{F}_{2^q} after suitable padding. Form the polynomial $A(X) := a_0 + \dots + a_{\rho\mathcal{N}} X^{\rho\mathcal{N}}$ and evaluate it in the first \mathcal{N} points of \mathbb{F}_{2^q} : $\forall i \in \{1, \dots, \mathcal{N}\} y_i := A(i)$.

4. Construct the augmented packets as:

$$\forall i \in \{1, \dots, \mathcal{N}\} \quad \text{AP}_i := \text{BID} \| i \| E_i \| N_i^1 \| \dots \| N_i^{d_i} \| y_i$$

Output: $\{\text{AP}_1, \dots, \text{AP}_{\mathcal{N}}\}$: set of augmented packets.

We first notice that, even when the channel rates α, β change, the structure of the LT code does not need to be modified since we keep working with the same inputs P_1, \dots, P_n and the same value c for the Robust Soliton distribution. Only the number \mathcal{N} of encoding symbols to be generated increases. This is an advantage over LTT and PRABS since the size of their field as well as the rate of their code have to be updated in case of modifications of network rates. In addition, it can be shown that the ratio $\frac{\mathcal{N}}{n}$ (as a function of n) is asymptotically bounded by a constant (see Appendix A). Notice that the list of irreducible polynomials over \mathbb{F}_2 is used at Step 3 when performing polynomial evaluations over \mathbb{F}_{2^q} . Those evaluations work as follows. Since any element of \mathbb{F}_{2^q} can be represented as $\lambda_0 Y^0 + \lambda_1 Y^1 + \dots + \lambda_{q-1} Y^{q-1}$ where each λ_i belongs to \mathbb{F}_2 , we define the first \mathcal{N} elements as $(0, \dots, 0)$, $(1, 0, \dots, 0)$, $(0, 1, 0, \dots, 0)$, $(1, 1, 0, \dots, 0)$ and so on until the binary decomposition of $\mathcal{N} - 1$.

The justification for the choice of q and the length of the pad of τ can be found in Appendix D.1. It should be noticed that the public values of Table 1 are sufficient to compute q and the pad length.

We now describe the algorithm run by the receivers to authenticate data they collected.

4.2 Security and Recovery of the Scheme

Security of the Scheme. We will now analyze the security of our authentication scheme. We want the receivers to authenticate data despite malicious actions performed by \mathcal{O} . Similarly to [35], we give the following definition:

Definition 1 *The collection of algorithms (KeyGen, Authenticator, Decoder) constitutes a secure and (α, β) -correct probabilistic multicast authentication scheme if no probabilistic polynomial-time opponent \mathcal{O} can win with a non-negligible probability the following game:*

- i) A key pair (SK, PK) is generated by KeyGen.
- ii) \mathcal{O} is given: (a) The public key PK and (b) Oracle access to Authenticator (but \mathcal{O} can only issue at most one query with the same block identification tag BID).
- iii) \mathcal{O} outputs $(\text{BID}, n, \alpha, \beta, \rho, \delta, \text{RP})$.

\mathcal{O} wins if one of the following happens:

- a) (violation of the correctness property) \mathcal{O} succeeds to output RP such that even if it contains $\lceil \alpha \mathcal{N} \rceil$ packets amongst a total not exceeding $\lfloor \beta \mathcal{N} \rfloor$ elements) of some authenticated packet set AP_i for block identification tag BID, decoding failure probability δ and parameters n, α, β, ρ , the decoder authenticates some incorrect packets.
- b) (violation of the security property) \mathcal{O} succeeds to output RP such that the decoder outputs $\{P'_1, \dots, P'_n\}$ which is non-empty and was never authenticated by Authenticator for the value BID, the probability δ and parameters n, α, β, ρ .

Algorithm 5 Decoder1

Input: The public key PK, the block number BID, Table 1 and the set of received packets RP.

1. Compute \mathcal{N} . Write the received elements as $\text{BID}_i \| j_i \| \tilde{E}_{j_i} \| \tilde{N}_{j_i}^1 \| \dots \| \tilde{N}_{j_i}^{d_{j_i}} \| \tilde{y}_{j_i}$ and discard those having $\text{BID}_i \neq \text{BID}$ or $j_i \notin \{1, \dots, \mathcal{N}\}$. Denote N the number of remaining packets. If $(N < \lceil \alpha n \rceil$ or $N > \lfloor \beta n \rfloor$) then the algorithm stops.
2. Rename the set of received packets $\{\widehat{\text{AP}}_1, \dots, \widehat{\text{AP}}_N\}$ and write each element as: $\widehat{\text{AP}}_i = \text{BID} \| j_i \| \tilde{E}_{j_i} \| \tilde{N}_{j_i}^1 \| \dots \| \tilde{N}_{j_i}^{d_{j_i}} \| \tilde{y}_{j_i}$ where $j_i \in \{1, \dots, \mathcal{N}\}$. Compute q as in Step 3 of Authenticator1. Get the irreducible polynomial of degree q from the sender's public list and run MPR on the set $\{(j_i, \tilde{y}_{j_i}), 1 \leq i \leq N\}$ to get a list $\{c_1, \dots, c_\mu\}$ of candidates for signature verification. If MPR rejects that set then the algorithm stops.
3. Initialize $h'_j := \emptyset$ for $j \in \{1, \dots, \mathcal{N}\}$ and $i := 1$. While the list has not been exhausted (and the signature not verified yet), pick c_i and write it as $h'_1 \| \dots \| h'_{\mathcal{N}} \| \sigma^i$ after removing the pad where each h'_k is \mathcal{H} bits long. If $\text{Verify}_{\text{PK}}(h(\text{BID} \| \tau_{\text{par}} \| h'_1 \| \dots \| h'_{\mathcal{N}}), \sigma^i) = \text{TRUE}$ then we set $h'_j = h'_j$ for $j \in \{1, \dots, \mathcal{N}\}$ and break out the loop. Otherwise, increment i by 1 and start again the while loop.
4. If $(h'_1, \dots, h'_{\mathcal{N}}) = (\emptyset, \dots, \emptyset)$ then the algorithm stops. Otherwise, set $E'_\lambda := \emptyset$ for $\lambda \in \{1, \dots, \mathcal{N}\}$. For each $\widehat{\text{AP}}_i$ written as at Step 2, if $h(\tilde{E}_{j_i} \| \tilde{N}_{j_i}^1 \| \dots \| \tilde{N}_{j_i}^{d_{j_i}}) = h_\lambda$ then $E'_\lambda = \tilde{E}_{j_i}$, $d'_\lambda = d_{j_i}$ and $\forall \xi \in \{1, \dots, d_{j_i}\} N'_\lambda{}^\xi = \tilde{N}_{j_i}^\xi$.
5. Pick the first $\lceil \alpha \mathcal{N} \rceil$ non-empty elements E'_μ and decode the LT code using the E'_μ 's as encoding symbols with degree d_μ and adjacent vertices positions $N'_\mu{}^1, \dots, N'_\mu{}^{d_\mu}$. Get n input symbols $\{P'_1, \dots, P'_n\}$ (where some of them can be empty).

Output: $\{P'_1, \dots, P'_n\}$: set of authenticated packets.

The difference from the definition given in [35] is that the packets are authenticated by the receiver with certain probability. In short, even if the receiver gets a set RP having at least $\lceil \alpha \mathcal{N} \rceil$ original elements, the whole original set $\{P_1, \dots, P_n\}$ is recovered with some probability. Nevertheless, Definition 1 involves that no incorrect packets can be authenticated. That is to say: $\forall i \in \{1, \dots, n\} P'_i \in \{\emptyset, P_i\}$ where P'_i denotes the i^{th} packet output by Decoder1. Lysyanskaya *et al.* showed that LTT was secure and (α, β) -correct. Following their arguments, we obtain the following result the proof of which is given in Appendix B.

Theorem 3 *The scheme (KeyGen, Authenticator1, Decoder1) is secure and (α, β) -correct.*

Thus, our authentication scheme is as secure and correct as LTT. We will now study the packet authentication probability of our protocol.

Recovery Property. We will now demonstrate that our scheme enables any receiver to recover the n data packets with a good probability and the number of signature verifications to be performed per block is $O(1)$ (as for LTT and PRABS). We also provide a non-asymptotic bound on this number of verifications. We now present our main theorem for this construction the proof of which is given in Appendix C.

Theorem 4 *Given the scheme (KeyGen, Authenticator1, Decoder1), for any BID, each receiver recovers the n original data packets P_1, \dots, P_n with probability at least $1 - \delta$. In addition, the number of signature verifications to be performed is upper bounded by:*

$$U(\mathcal{N}) := \min(\lfloor U_1(\mathcal{N}) \rfloor, \lfloor U_2(\mathcal{N}) \rfloor)$$

where:

$$\begin{cases} U_1(\mathcal{N}) = \frac{1}{\rho \mathcal{N}} \left(\frac{1}{\sqrt{\alpha^2 - \beta \rho}} - 1 \right) + \frac{\beta}{\alpha^2 - \beta \rho} + \frac{1}{\rho} \\ U_2(\mathcal{N}) = \frac{\beta}{2(\alpha^2 - \beta \rho)} + \frac{1}{\rho} + \frac{\sqrt{\beta^2 + \frac{4}{\rho^2 \mathcal{N}^2} (1 - \rho \alpha)}}{2(\alpha^2 - \beta \rho)} - \frac{1}{\rho \mathcal{N}} \end{cases}$$

which is $O(1)$ as a function of the block length n .

4.3 Other Families of Rateless Codes

In this section, we will compare the complexity in encoding/decoding of LT, Online and Raptor codes. Indeed, the security, correctness and recovery property of our scheme only depend on the fact that the LT decoding algorithm is consistent which is also the case for Online and Raptor codes. In addition, we will also compare these families to the modified LT codes introduced by Harrelson *et al.* [25]. In their work, they changed the construction of LT codes given by Luby [32] to

fit them to their practical implementations without altering their optimality (i.e. if we generate enough symbols then we can have $\delta \simeq 0$). Their technique consists of modifying the way the neighbors of each encoding symbol E are chosen. As in [32], the degree d is chosen using the Robust Soliton distribution. Instead of uniformly choosing the d neighbors, Harrelson *et al.* proposed to uniformly choose two integers a and b and to generate the positions of the d neighbors as $a i + b$ for $i \in \{1, \dots, d\}$. Thus, it is useless to append the neighbors to the encoding symbol for transmission since only $E\|a\|b\|d$ needs to be sent. This means that the overhead per encoding symbol has a fixed and much smaller size than in [32]. This is of particular interest in our case (Step 4 of Authenticator1) since our overhead per packet is particularly limited and such a fixed size helps to avoid data congestion due to irregular flow of information within the network.

Contrary to block codes which use finite field operations to encode and decode data, these families of rateless codes rely on XOR operations over packets. Based on the work done in [25, 32, 37, 60], we built Table 2. Both Raptor and Online codes require preprocessing of data before encoding. In [37], Maymounkov proposed two different ways to do so for Online codes. The complexities shown in Table 2 correspond to the second method since the first technique involves a dependence between the packet authentication probability and the number of packets per block. The notation ϵ_δ means that the element depends on the decoding failure probability δ but is independent from n .

	Average number XOR operations for decoding	Number of encoding symbols generated (m)	Decoding failure probability	Encoding symbol overhead
LT codes	$O(n \log(n/\delta))$	$n + O(\sqrt{n} \log^2(n/\delta))$	δ	variable
LT codes (modified)	$O(n \log(n/\delta))$	$n + O(n^{5/6} \text{polylog}(n, 1/\delta))$	δ	constant
Online codes	$O(n \log(1/\epsilon_\delta))$	$(1 + \epsilon_\delta)n$ (fixed $\epsilon_\delta > 0$)	$O(\delta^\eta)$ (fixed $\eta > 0$)	variable
Raptor codes	$O(n \log(1/\epsilon_\delta))$	$(1 + \epsilon_\delta)n$ (fixed $\epsilon_\delta > 0$)	δ	variable

Table 2: Complexity comparison for different classes of rateless codes.

According to Table 2, Online and Raptor codes seem to have better encoding and decoding complexities than LT codes. Nevertheless, Raptor codes were designed for the Binary Erasure channel (BEC) since the efficiency of its preprocessing part relies on the existence on good pre-codes to achieve linear time for both encoding and decoding process. That is, the property which is achieved by Tornado codes on BEC [34, 60]. Given our opponent model, it is unlikely that BEC accurately model the actions performed by \mathcal{O} . Nevertheless, a recent work by Palanki and Yedidia [44] suggests that Raptor codes can still be practically more efficient than LT codes for our authentication scheme. Indeed, they implemented both classes of codes on Additive White Gaussian Noise Channel and Binary Symmetric Channel and noticed that, even on these channels, Raptor codes outperformed LT codes for decoding. Etesami *et al.* [15] performed analogous implementations and their results exhibited the same behavior. They also showed that Raptor codes could perform quite well on any arbitrary symmetric channel. This behavior was confirmed in [16, 29, 33]. The performance of Raptor codes over other communication channels has been studied in [53, 66]. A recent survey by Demir and Aktaş showed that Raptor codes also outperformed Reed-Solomon codes (as used in LTT) for some multimedia applications [14].

As suggested by Harrelson *et al.* [25], it is possible to reduce the size of information to be transmitted and achieve a regular packet overhead at the cost of extra symbols for decoding (see Table 2). Since achieving a uniform throughput within the communication channel avoids data congestion, substituting original LT codes by their modifications in our authentication protocol is recommended. Since Raptor codes are the concatenation of an erasure code (as Tornado codes for instance) and a LT code, these modifications can also be applied to these codes. Therefore, we believe that practical implementations of the authentication scheme described in Section 4 will be even more efficient when substituting LT codes by Raptor codes (exhibiting the same modifications for their internal LT coding).

As proved in Section 4.2, the value $\mathcal{N} = \lceil \frac{m}{\alpha} \rceil$ represents the number of encoded symbols the sender has to create so that each receiver is guaranteed to obtain at least m of them. This enables each receiver to recover the n data packets P_1, \dots, P_n with probability at least $1 - \delta$. Nevertheless, the threshold values m depicted in Table 2 can be too large for some applications. Indeed, the larger m is, the larger the number of elements to be stored at the receiver and the longer the time spent by the sender to build the augmented packets are. Karp *et al.* [28] gave a formula expressing the probability of non-decoding u packets amongst n after receiving a fixed value of encoding symbols which can be chosen by the sender. This can be useful if the application which will run the received packets has a tolerance rate for loss of content. In such a situation, the sender computes the number $\mathcal{N}_u (< \mathcal{N})$ of packets he has to transmit in order to achieve at most this rate of non-recovered packets reducing his processing time and the storage requirement at the receiver.

5 MDS Codes for Multicast Stream Authentication

In this section, we will describe a multicast authentication protocol using MDS codes which is robust against packet loss and data injection. As in Section 4, our technique allows any new user to join the communication group at any block boundary. We will demonstrate its security, recovery property and show that the number of signature verifications per block at the receiver is $O(1)$ as a function of the block length n . Finally, we justify our choice for Lacan and Fimes' construction.

5.1 Our Authentication Protocol

The values n, α, β and ρ are the same as in Section 4.1. In particular, we assume that ρ is uniquely determined when n, α and β are known. In order to have $\lceil \alpha n \rceil$ symbols of equal length for encoding, we must pad $P_1 \parallel \dots \parallel P_n$ with $\tilde{\ell}$ zeros appropriately (see Step 1 of Algorithm 6). Then, we can split $P_1 \parallel \dots \parallel P_n \parallel 0^{\tilde{\ell}}$ into n symbols $S_1, \dots, S_{\lceil \alpha n \rceil}$ where each S_i is \tilde{q} bits long with $\tilde{q} := \left\lceil \frac{n\mathcal{P}}{\lceil \alpha n \rceil} \right\rceil$. The efficiency parameter r is chosen by the sender as a divisor of \tilde{q} such that the receivers can efficiently perform computations over the field \mathbb{F}_{2^s} where $s := \frac{\tilde{q}}{r}$. Notice that the smaller r is, the larger the number of messages to be encoded at Step 2 of EncodeMDS is. Table 3 summarizes the scheme parameters which are assumed to be publicly known. We denote τ_{par} the tag representing the communication parameters, namely: $\tau_{\text{par}} := n \parallel \alpha \parallel \beta \parallel \mathcal{P}$.

n : Block length	r : Efficiency parameter of the MDS code
α : Survival rate	β : Flood rate
\mathcal{P} : Packet size (in bits)	G : Generating matrix of the MDS code
A list of irreducible polynomials over \mathbb{F}_2	

Table 3: Public parameters for our MDS code-based scheme.

Algorithm 6 Authenticator2

Input: The secret key SK, the block number BID, Table 3 and n data packets P_1, \dots, P_n .

1. Compute: $\tilde{b} = n\mathcal{P} \bmod \lceil \alpha n \rceil$. Denote $\tilde{\ell}$ as $\tilde{\ell} := (0 \text{ if } \tilde{b} = 0) \text{ or } (\lceil \alpha n \rceil - \tilde{b} \text{ otherwise})$. Write $P_1 \parallel \dots \parallel P_n \parallel 0^{\tilde{\ell}}$ as $S_1 \parallel \dots \parallel S_{\lceil \alpha n \rceil}$ where each S_i is $\tilde{q} := \left\lceil \frac{n\mathcal{P}}{\lceil \alpha n \rceil} \right\rceil$ bits long.
2. Pick the polynomial $\tilde{Q}(X)$ of degree $s = \frac{\tilde{q}}{r}$ from the list. Compute $(\hat{S}_1, \dots, \hat{S}_n) = \text{EncodeMDS}(n, \lceil \alpha n \rceil, \tilde{Q}(X), G, S_1, \dots, S_{\lceil \alpha n \rceil}, r)$.
3. Compute: $\forall i \in \{1, \dots, n\} h_i = h(\hat{S}_i)$ and form the authentication tag $\tau := h_1 \parallel \dots \parallel h_n \parallel \sigma$ where σ is computed as: $\sigma = \text{Sign}_{\text{SK}}(h(\text{BID} \parallel \tau_{\text{par}} \parallel h_1 \parallel \dots \parallel h_n))$.
4. Denote ξ the smallest element of \mathbb{N} such that:

$$\left\lceil \frac{\mathcal{H}n + \mathcal{S} + \xi}{\rho n + 1} \right\rceil \geq \lceil \log_2 n \rceil \quad (2)$$

Denote q the left hand side of Inequality (2). Write τ as the concatenation $a_0 \parallel \dots \parallel a_{\rho\mathcal{N}}$ of $(\rho\mathcal{N} + 1)$ elements of \mathbb{F}_{2^q} after suitable padding. Form the polynomial $A(X) := a_0 + \dots + a_{\rho\mathcal{N}} X^{\rho n}$ and evaluate it in the first n points of \mathbb{F}_{2^q} : $\forall i \in \{1, \dots, n\} y_i := A(i)$.

5. Construct the augmented packets as:

$$\forall i \in \{1, \dots, n\} \quad \text{AP}_i := \text{BID} \parallel i \parallel \hat{S}_i \parallel y_i$$

Output: $\{\text{AP}_1, \dots, \text{AP}_n\}$: set of augmented packets.

The value of q as well as the pad occurring at Step 4 can be found in Appendix D.2. For our construction, we can assume, without loss of generality, that, when the field, length and dimension of the MDS code are known then the couple (G, r) is unique so that knowing $(n, \alpha, \beta, \mathcal{P})$ (representing τ_{par}) is enough to determine each step of Authenticator2.

Upon reception of data, the receivers use Algorithm 7 to authenticate information.

5.2 Security and Recovery of the Scheme

Security of the Scheme. Similarly to [35], we give the following definition:

Algorithm 7 Decoder2

Input: The public key PK, the block number BID, Table 3 and the set of received packets RP.

1. Write the packets as $\text{BID}_i \| j_i \| \tilde{S}_{j_i} \| \tilde{y}_{j_i}$ and discard those having $\text{BID}_i \neq \text{BID}$ or $j_i \notin \{1, \dots, n\}$. Denote N the number of remaining elements. If $(N < \lceil \alpha n \rceil$ or $N > \lfloor \beta n \rfloor$) then the algorithm stops.
2. Rename the remaining elements as $\{\tilde{\text{AP}}_1, \dots, \tilde{\text{AP}}_N\}$ and write each element as: $\tilde{\text{AP}}_i = \text{BID} \| j_i \| \tilde{S}_{j_i} \| \tilde{y}_{j_i}$ where $j_i \in \{1, \dots, n\}$. Compute q as in Step 3 of Authenticator2. Get the irreducible polynomial of degree q from the sender's public list and run MPR on the set $\{(j_i, \tilde{y}_{j_i}), 1 \leq i \leq N\}$ to get a list $\{c_1, \dots, c_\mu\}$ of candidates for signature verification. If MPR rejects that set then the algorithm stops.
3. Initialize $h'_j = \emptyset$ for $j \in \{1, \dots, n\}$. While the list has not been exhausted (and the signature not verified yet), pick c_i and write it as: $h_1^i \| \dots \| h_n^i \| \sigma^i$ after removing the pad where each h_k^i is \mathcal{H} bits long. If $\text{Verify}_{\text{PK}}(h(\text{BID} \| \tau_{\text{par}} \| h_1^i \| \dots \| h_n^i), \sigma^i) = \text{TRUE}$ then set $h'_j = h_j^i$ for $j \in \{1, \dots, n\}$ and break out the loop. Otherwise, increment i by 1 and start again the while loop.
4. If $(h'_1, \dots, h'_n) = (\emptyset, \dots, \emptyset)$ then the algorithm stops. Otherwise, set $\hat{S}'_k := \emptyset$ for all $k \in \{1, \dots, n\}$. For each $\tilde{\text{AP}}_i$ written as at Step 2, if $h(\tilde{S}_{j_i}) = h_\lambda$ then $\hat{S}'_\lambda = \tilde{S}_{j_i}$.
5. If we have less than $\lceil \alpha n \rceil$ non-empty symbols then the algorithm stops. Otherwise, denote $\hat{S}'_{p_1}, \dots, \hat{S}'_{p_\gamma}$ the non-empty elements. Decode them as: $(S'_1, \dots, S'_{\lceil \alpha n \rceil}) = \text{DecodeMDS}(n, \lceil \alpha n \rceil, \tilde{Q}(X), G, (p_1, \hat{S}'_{p_1}), \dots, (p_\gamma, \hat{S}'_{p_\gamma}), r)$ after computing \tilde{q} and choosing $\tilde{Q}(X)$ as in Authenticator2.
6. Remove the pad from $S'_1 \| \dots \| S'_{\lceil \alpha n \rceil}$ and write the remaining string as $P'_1 \| \dots \| P'_n$ where each P'_i is \mathcal{P} bits long.

Output: $\{P'_1, \dots, P'_n\}$: set of authenticated packets.

Definition 2 *The collection of algorithms (KeyGen, Authenticator, Decoder) constitutes a secure and (α, β) -correct multicast authentication scheme if no probabilistic polynomial-time opponent \mathcal{O} can win with a non-negligible probability the following game:*

- i) A key pair (SK, PK) is generated by KeyGen.
- ii) \mathcal{O} is given: (a) The public key PK and (b) Oracle access to Authenticator (but \mathcal{O} can only issue at most one query with the same block identification tag BID).
- iii) \mathcal{O} outputs $(\text{BID}, n, \alpha, \beta, \rho, \mathcal{P}, \mathcal{Q}(X), \tilde{Q}(X), G, r, \text{RP})$.

\mathcal{O} wins if one of the following happens:

- a) (violation of the correctness property) \mathcal{O} succeeds to output RP such that even if it contains $\lceil \alpha N \rceil$ packets (amongst a total not exceeding βn elements) of some authenticated packet set AP_i for block identification tag BID and parameters $n, \alpha, \beta, \rho, \mathcal{P}$, the decoder fails to authenticate all the correct packets.
- b) (violation of the security property) \mathcal{O} succeeds to output RP such that the decoder outputs $\{P'_1, \dots, P'_n\}$ that was never authenticated by Authenticator for the value BID and parameters $n, \alpha, \beta, \rho, \mathcal{P}$.

The difference between Definition 1 and Definition 2 is that the latter requires total recovery of all data elements. In Definition 1, Points a) and b) mean that the decoder cannot output packets P'_j such that $P'_j \notin \{\emptyset, P_j\}$. This is the same definition as for LTT. Our new definition stipulates that it cannot output packets $P'_j \neq P_j$. This is due to the fact that we use an erasure correcting code allowing complete recovery of the stream. Definition 2 can be seen as the generalization of Definition 1 when the decoding failure probability δ is 0. As a consequence, if an authentication scheme (KeyGen, Authenticator, Decoder) is secure and correct as specified by Definition 2 then it is also secure and correct according to Definition 1. We have the following result for our MDS code-based protocol.

Theorem 5 *Our scheme (KeyGen, Authenticator2, Decoder2) is secure and (α, β) -correct.*

The proof is analogous to the proof of Theorem 3 and omitted.

Recovery Property. We will now demonstrate that our scheme enables any receiver to recover the n data packets (as in [27]) and the number of signature verifications to be performed per block is $O(1)$ (as in [27, 35]). As in Section 4.2, we provide a non-asymptotic bound on the number of verifications.

Theorem 6 *Given the scheme (KeyGen, Authenticator2, Decoder2), for any BID, each receiver recovers the n original data packets P_1, \dots, P_n . In addition, the number of signature verifications to be performed is upper bounded by $U(n)$ (where the function $U(\cdot)$ is defined in Theorem 4) which is $O(1)$ as a function of the block length n .*

The proof is given in Appendix C. It should be noticed that this bound is also valid for LTT.

5.3 Coding Complexity

The field used by the MDS code is \mathbb{F}_{2^s} where s is defined as in Section 5.1. Our results are based on the analysis done in [30]. Encoding a single message m_i requires $O(n \log n)$ field operations. Therefore, encoding the r messages needs $O(r n \log n)$ field operations. Similarly, decoding a single c'_i requires $O(n \log^2 n)$ field operations. Since r of them must be decoded, we get $O(r n \log^2 n)$ as total decoding complexity. It is clear that $r \in O(1)$ as a function of n . So, the previous two complexities become $O(n \log n)$ and $O(n \log^2 n)$, respectively.

In [27], it was suggested to use Reed-Solomon codes as erasure codes for PRABS. Since Reed-Solomon codes are MDS [36], it is natural to compare their efficiency to our MDS code's one. Notice that the messages processed by PRABS's code have the same number of symbols than ours but these symbols are taken from a smaller field than for our construction. In our survey, however, we focus on the number of field operations to encode/decode data as it is the standard tool in coding theory to compare the efficiency of two coding techniques since a fair comparison presumes that both codes are used over the same field.

In their work, Karlof *et al.* referred to the original paper by Reed and Solomon [55] to encode/decode data. Based on this consideration encoding/decoding a single m_i/c'_j requires $O(n^2)$ field operations. Then, it is clear that our complexity for encoding/decoding are better. However, Reed-Solomon codes can be processed in a more efficient way than in [55] using the technique developed by Guruswami and Sudan in [24]. This technique which was used in [35] enables any message m_i to be encoded in $O(n \log^2 n)$ field operations and any codeword c'_j to be decoded using $O(n^2)$ field operations. If their technique is to be used to encode/decode data in PRABS then the encoding complexity of our protocol represents an increase by factor $\log n$ (for the whole set of r messages) whereas the decoding complexity is reduced by a factor $\frac{\log^2 n}{n}$ (for the whole set of r codewords). We claim that, even in this case, the code we employed still gives more benefits than PRABS's one. Indeed, in our network model the sender is assumed to be more computationally powerful than the receivers. Therefore, he can cope with the small increasing factor $\log n$ since n will be roughly 1000 in practical applications. At the same time, the receivers take advantage of the complexity reduction from quadratic (using Reed-Solomon codes) to sub-quadratic (with our technique).

After treating the case of Reed-Solomon codes, it remains to argue that our MDS code construction gives better complexity for encoding/decoding than other MDS codes used in practical implementations. In [30], Lacan and Fimes pointed out that in computer communication, erasure codes were used in systematic form. They also emphasized that systematic MDS codes employed in practical applications relied on either Cauchy or Vandermonde matrices to generate the redundancy symbols. They proved that their code construction exhibited better complexity for both encoding and decoding than any other systematic MDS code relying on either matrix construction.

Based on these observations, we claim that using Lacan and Fimes' codes for our authentication scheme gives us an optimal erasure correcting technique for data streaming over a multicast network.

6 Comparison of Schemes

In this section, we compare our two authentication schemes. We survey three critical points: the signature complexity, the packet overhead and the computational cost of our erasure correcting codes.

6.1 Signature Complexity

In Section 5.2, we explained that $U(n)$ was also a bound for LTT. According to Theorem 3 from [27] the number of signature verifications for PRABS is upper bounded by $V(n) := \left\lfloor \frac{\lfloor \beta n \rfloor - \lceil \alpha n \rceil}{\lfloor \alpha n \rfloor} \right\rfloor + 1 = \left\lfloor \frac{\lfloor \beta n \rfloor}{\lfloor \alpha n \rfloor} \right\rfloor$. Since $V(n) \leq \frac{\beta}{\alpha}$, it is clear that $V(n)$ is $O(1)$ and $V(n) \leq \lfloor U_1(n) \rfloor$. We compare the uppers bounds on the number of signature verifications to be performed per block in Table 4. As in [51], we chose $n = 1000$. We considered ρ equal to 10%, 30%, 50%, 70% and 90% of the threshold $\frac{\alpha^2}{\beta}$ for each couple (α, β) . The reader may notice that some values 1000ρ are not integers (as it should be for our scheme). We committed this abuse because our main goal was to exhibit the behavior of $U(n)$ for a realistic value n . If we had been to be consistent with the fact that ρn is an integer then the smallest integer n valid for all couples (α, β) in our table should have been 132,000 which is not a realistic assumption for the block length n . In addition, we set the value c for the LT code as the middle of the allowed interval, i.e.:

$$c = \frac{1}{2 \ln(n/\delta)} \left(\frac{n^n}{2} + \frac{\sqrt{n}}{n-1} \right)$$

and picked $\delta = 0.1$ and $\eta = \frac{1}{4}$.

Notice that, as claimed in Section 2.2, c is independent of both rates α and β . Thus, it is not affected by variations of those values which provides stability for the code structure throughout the communication process.

		(α, β)							
		(0.5, 1.1)	(0.5, 1.25)	(0.5, 1.5)	(0.5, 2)	(0.75, 1.1)	(0.75, 1.25)	(0.75, 1.5)	(0.75, 2)
10%		48 48 2	55 55 2	66 66 3	88 88 4	21 21 1	24 24 1	29 29 2	39 39 2
30%		20 20 2	23 23 2	28 28 3	38 38 4	9 9 1	10 10 1	12 12 2	16 16 2
50%		17 17 2	19 19 2	23 23 3	31 31 4	7 7 1	8 8 1	10 10 2	14 14 2
70%		20 20 2	23 23 2	28 28 3	38 38 4	9 9 1	10 10 1	12 12 2	16 16 2
90%		48 48 2	55 55 2	66 66 3	88 88 4	21 21 1	24 24 1	28 28 2	36 36 2
		(α, β)							
		(0.8, 1.1)	(0.8, 1.25)	(0.8, 1.5)	(0.8, 2)	(0.9, 1.1)	(0.9, 1.25)	(0.9, 1.5)	(0.9, 2)
10%		19 19 1	21 21 1	26 26 1	34 34 2	15 15 1	17 17 1	20 20 1	27 27 2
30%		8 8 1	9 9 1	11 11 1	14 14 2	6 6 1	7 7 1	8 8 1	11 11 2
50%		6 6 1	7 7 1	9 9 1	12 12 2	5 5 1	6 6 1	7 7 1	9 9 2
70%		8 8 1	9 9 1	10 10 1	14 14 2	6 6 1	7 7 1	8 8 1	11 11 2
90%		19 19 1	21 21 1	26 26 1	34 34 2	15 15 1	17 17 1	20 20 1	27 27 2

Table 4: Upper bounds for our LT code-based scheme, our MDS code-based scheme and PRABS.

We first notice that $U(\mathcal{N})$ and $U(n)$ are identical for our choice of parameters. This can be explained by the fact that \mathcal{N} and n only interfere via denominators. Therefore, the behavior of $U_1(\mathcal{N})$ and $U_1(n)$ is controlled by $\frac{\beta}{\alpha^2 - \beta\rho} + \frac{1}{\rho}$ while $U_2(\mathcal{N})$ and $U_2(n)$ mainly depend on $\frac{\beta}{2(\alpha^2 - \beta\rho)} + \frac{1}{\rho}$. Table 4 also clearly shows that $V(n)$ is much smaller than $U(n)$ (and $U(\mathcal{N})$). Nevertheless, this low value of $V(n)$ is precisely due to the fact that each augmented packet carries $\log n$ hashes since the hashes are used to partition the received elements into (at most) $V(n)$ sets. Thus, a logarithmic number of hashes per augmented packet is the price paid by Karlof *et al.* to achieve low number of signature verifications. As noted earlier, their approach based on Merkle hash trees is impractical since such large packets can cause congestion in the network throughput. Table 4 results suggest that $U(n)$ is minimal when ρ is roughly half the threshold value $\frac{\alpha^2}{\beta}$.

6.2 Packet Overhead

The packet overhead is the length of the extra tag of information used to provide authentication. Notice that an augmented packet without a tag is assumed to be written as: $\text{BID}||i||P_i$. Remember that the bit size of packets $P_{i,j}$ is \mathcal{P} .

6.2.1 LT Code-based Scheme

Its augmented packets are written as:

$$\text{BID}||i||E_i||N_i^1||\dots||N_i^{d_i}||y_i$$

where E_i has the same length as the original packets, each $N_i^j \in \{1, \dots, n\}$ and $y_i \in \mathbb{F}_{2^q}$. So, our packet overhead is:

$$d_i (\lceil \log_2 n \rceil + 1) + \left\lceil \frac{\mathcal{H}\mathcal{N} + \mathcal{S} + \xi}{\rho\mathcal{N} + 1} \right\rceil$$

As d_i varies from symbol to symbol, this yields to an irregular overhead. However, if we use the modified version of LT codes proposed by Harrelson *et al.* [25], then the augmented packets are written as $\text{BID}||i||E_i||a||b||d_i||y_i$, where $a, b, d_i \in \{1, \dots, n\}$ (see Section 4.3). In that case, the bit size of the overhead is:

$$3 (\lceil \log_2 n \rceil + 1) + \left\lceil \frac{\mathcal{H}\mathcal{N}' + \mathcal{S} + \xi'}{\rho\mathcal{N}' + 1} \right\rceil,$$

where $\mathcal{N}' := \lceil \frac{m'}{\alpha} \rceil$ with $m' (> m)$ being the new bound guaranteeing decoding failure with probability at most δ and ξ' is the smallest element of \mathbb{N} such that:

$$\left\lceil \frac{\mathcal{H}\mathcal{N}' + \mathcal{S} + \xi'}{\rho\mathcal{N}' + 1} \right\rceil \geq \lceil \log_2 \mathcal{N}' \rceil$$

Based on the work by Harrelson *et al.*, we deduce that m' is equal to:

$$\left\lceil \frac{A \log_2(n/\delta)}{n} + \frac{M (\log_2 n)^3 n^2}{36 A^3 B^2} + n^{9-M/2} + 4B \right\rceil,$$

where:

$$\begin{aligned} M &:= \max(20, 2(8 - \log_2(\delta/6))/\log_2 n) \\ A &:= C_A n^{5/6} (\log_2 n)^{1/2} (\log_2(n/\delta))^{-1/2} M^{1/6} \\ B &:= C_B n^{-1/6} (\log_2 n)^{1/2} (\log_2(n/\delta))^{1/2} M^{1/6} \end{aligned}$$

provided that the elements C_A and C_B verify that:

$$\begin{aligned} C_B &< \frac{n^{1/6}}{4 (\log_2 n)^{1/2} (\log_2(n/\delta))^{1/2} M^{1/6}} \\ C_B^2 C_A^3 &\geq \frac{e}{36} \frac{M^{1/3} \log_2 n \log_2(n/\delta)}{n^{1/3}} \end{aligned}$$

6.2.2 MDS Code-based scheme

The augmented packets are written as:

$$\text{BID} \| i \| \hat{S}_i \| y_i$$

The bit size of the element y_i is $\left\lceil \frac{n\mathcal{H} + \mathcal{S} + \xi}{n\rho + 1} \right\rceil$ (as for LTT) and \hat{S}_i is slightly larger than P_i since it is $\left\lceil \frac{n\mathcal{P}}{\lceil \alpha n \rceil} \right\rceil$ bits long. So, our packet overhead is:

$$\left\lceil \frac{n\mathcal{P}}{\lceil \alpha n \rceil} \right\rceil - \mathcal{P} + \left\lceil \frac{n\mathcal{H} + \mathcal{S} + \xi}{n\rho + 1} \right\rceil$$

Notice that the difference $\left\lceil \frac{n\mathcal{P}}{\lceil \alpha n \rceil} \right\rceil - \mathcal{P}$ comes from the fact that we use an code correcting up to a fraction $1 - \alpha$ of erasures.

6.2.3 Comparison

We showed above that our augmented packet size is slightly larger than those from LTT. We would like to draw the reader's attention to an important fact. When studying the packet overhead of their scheme, Lysyanskaya *et al.* claimed that it was $\frac{\mathcal{H}}{\rho} + O(1)$ bits long. Unfortunately, this is incorrect. Indeed, as in our construction, the field \mathbb{F}_{2^q} must contain at least n points for polynomial evaluation. Therefore, their construction also requires: $n \leq 2^q$ which demonstrate that their packet overhead is $\Omega(\log_2 n)$. In fact, it is easy to see that the packet overhead for LTT is $\Theta(\log_2 n)$ like our construction due to the choice of the integer ξ .

PRABS augmented packets are written as:

$$P_i \| s'_i \| w_i$$

where w_i is the concatenation of $\lceil \log_2 n \rceil$ hashes and s'_i is a field element of the MDS code used to correct up to a fraction $1 - \alpha$ of erasures [27]. The message to be encoded by PRABS is $\text{BID} \| h(P_1) \| \dots \| h(P_n) \| \sigma$. If we do not take into account packet numbering (whose size is negligible with respect to the rest of the string) then the bit size of the field elements used in PRABS is $\left\lceil \frac{n\mathcal{H} + \mathcal{S}}{\lceil \alpha n \rceil} \right\rceil$ which leads to a packet overhead for PRABS to be approximately:

$$\left\lceil \frac{n\mathcal{H} + \mathcal{S}}{\lceil \alpha n \rceil} \right\rceil + \lceil \log_2 n \rceil \mathcal{H}$$

We chose $n, \alpha, \beta, \delta, \eta$ as in Table 4. We also picked ρ as half the threshold value $\frac{\alpha^2}{\beta}$. We considered the (heuristically) collision resistant hash function SHA-256, a 1024-bit RSA signature (see Section 4.1) and the packets size \mathcal{P} as 64 bytes as in [45, 50]. With the values, the parameters C_A and C_B must verify:

$$\begin{aligned} C_B &< 0.0416 \\ C_B^2 C_A^3 &\geq 2.7142 \end{aligned}$$

We chose $C_A = 0.04$ and $C_B = 11.93$. Our results are shown in Table 5.

6.3 Coding Efficiency

In Section 5.3, we saw that the MDS code construction by Lacan and Fimes required $O(n \log_2 n)$ field operations for encoding and $O(n \log_2^2 n)$ field operations for decoding. The field of computations is \mathbb{F}_{2^s} where $s \in O(\hat{q})$. As any field operation over \mathbb{F}_{2^s} can be implemented in $O(s^2)$ bit operations [38], the complexity of encoding is $O(n s^2 \log_2 n)$ bit operations while decoding can be achieved in $O(n s^2 \log_2^2 n)$ bit operations.

		α											
		0.5			0.75			0.8			0.9		
β	1.1	2266	2755	3075	1032	1173	2903	912	1009	2882	728	754	2846
	1.25	2566	3057	3075	1167	1309	2903	1031	1129	2882	823	848	2846
	1.5	3063	3560	3075	1391	1535	2903	1228	1328	2882	979	1006	2846
	2	4048	4560	3075	1837	1986	2903	1621	1725	2882	1291	1321	2846

Table 5: Bit size of the packet overhead for our LT code-based scheme, our MDS code-based scheme and PRABS.

Harrelson *et al.* showed that the average number of packet XOR-ings was $O(\mathcal{N} \log(\mathcal{N}/\delta))$ for both encoding and decoding. Notice that this value is also valid for the original LT codes designed by Luby [32]. Since each XOR-ing requires \mathcal{P} bit operations, the total number of bit operations for our rateless code based scheme is $O(\mathcal{N} \log(\mathcal{N}/\delta) \mathcal{P})$.

The main difference between these two approaches rely on the bit complexity of basic operations which is linear in the input size for the rateless code construction while it becomes quadratic for the MDS code scheme. Therefore if the receivers have limited computational power then the first approach is to be privileged.

We are aware of the existence of linear time encoding/decoding (non-linear) codes [3, 22, 23, 26, 58]. We did not use them for our protocol because their construction parameters were not flexible enough to fit our network parameters. In addition, these codes are not MDS which involves extra-generation of symbols to achieve the same capacity of correction. Thus, the use of such erasure correcting codes would lead to a larger packet overhead than in Section 6.2 and Table 5.

7 Conclusion

In this paper, we presented two multicast stream authentication protocols which can be considered as extensions of LTT and PRABS. Our constructions provide non-repudiation of the sender and enable new members to join the communication group at any block boundary. Unlike LTT, our technique allows recovery of the original data packets. In addition, only $O(1)$ signature verifications are performed per block. At the same time, our packet overheads are lower than PRABS's which reduces the risk of network congestion. When performing video or audio streaming for instance, the recovery property can be used to prevent audio gaps or frozen images when playing the stream content. We also derived a non-asymptotic upper bound on the number of signature verifications to be performed which can be used in practice to obtain an upper bound on the authentication delay our schemes exhibit.

We proposed to use two different classes of codes. Our first approach, relying on rateless codes, exhibits a low complexity arithmetic while keeping reasonable overhead and thus represents an interesting solution for end-users with limited computational power. Our second construction is based on block codes with the following advantages. First, MDS codes, by definition, are optimal for correcting erasures. Second, the MDS code construction by Lacan and Fimes was proved in [30] to have a better complexity than most MDS codes used in practice. When compared to Reed-Solomon codes (as used in [35]), our erasure code encoding exhibited a slightly larger complexity which can however be compensated by the computational capacities of the sender. At the same time, the decoding complexity is much smaller than Reed-Solomon one which benefits to all receivers. We would like to point out that the discovery of faster encoding/decoding codes could improve the performance of our schemes.

Another point deserving investigation is the development of fast and space efficient accumulators as any scheme relying on such primitives will exhibit $V(n)$ signature verifications per block like PRABS which is quite small (see Table 4).

Finally, it is worth studying alternative techniques to digital signatures such as trapdoor hash functions [59] for instance. Indeed, in [62], Very Smooth Hash [12] was used to reduce the time needed to exhaust the list of size $U(n)$ output by MPR.

Acknowledgement

The authors are grateful to the anonymous reviewers for their comments to improve the quality of this paper. This work was supported by the Australian Research Council under ARC Discovery Projects DP0558773, DP0665035 and DP0663452. This work was also supported in part by the National Natural Science Foundation of China Grant 60553001 and the National Basic Research Program of China Grant 2007CB807900, 2007CB807901. Christophe Tartary did most of this work while at Macquarie University where his research was supported by an iMURS scholarship. The research of Huaxiong Wang is partially supported by the Ministry of Education of Singapore under grant T206B2204.

References

- [1] 3GPP TS 26.346 V7.2.0. Technical specification group services and system aspects; Multimedia Broadcast/Multimedia Service (MBMS); protocols and codecs. Available online at: <http://www.3gpp.org/ftp/Specs/html-info/26346.htm>, December 2006.
- [2] Mohamed Al-Ibrahim and Josef Pieprzyk. Authenticating multicast streams in lossy channels using threshold techniques. In *ICN 2001*, volume 2094 of *Lecture Notes in Computer Science*, pages 239 – 249, Colmar - France, July 2001. Springer - Verlag.
- [3] Noga Alon, Jeff Edmonds, and Michael Luby. Linear time erasure codes with nearly optimal recovery (extended abstract). In *FOCS'95*, pages 512 – 519, Milwaukee - USA, October 1995.
- [4] Krishna B. Athreya and Soumendra N. Lahiri. *Measure Theory and Probability Theory*. Springer, 2006.
- [5] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology - Eurocrypt'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480 – 494, Konstanz - Germany, May 1997. Springer - Verlag.
- [6] Paulo S. L. M. Barreto, Hae Y. Kim, Ben Lynn, and Michael Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology - Crypto'02*, volume 2442 of *Lecture Notes in Computer Science*, pages 354 – 369, Santa Barbara, USA, August 2002. Springer - Verlag.
- [7] Josh Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Advances in Cryptology - Eurocrypt'93*, volume 765 of *Lecture Notes in Computer Science*, pages 274 – 285, Lofthus, Norway, May 1993. Springer - Verlag.
- [8] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297 – 319, September 2004.
- [9] John W. Byers, Michael Luby, and Michael Mitzenmacher. A digital fountain approach to asynchronous reliable multicast. *IEEE Journal on Selected Areas in Communications*, 20(8):1528 – 1540, October 2002.
- [10] Pasquale Cataldi, Miquel Pedròs Shatarski, Marco Grangetto, and Enrico Magli. Implementation and performance evaluation of LT and Raptor codes for multimedia applications. In *IIH-MSP'06*, pages 263 – 266, Pasadena, USA, December 2006. IEEE Computer Society.
- [11] Yacine Challal, Hatem Bettahar, and Abdelmajid Bouabdallah. A taxonomy of multicast data origin authentication: Issues and solutions. *IEEE Communications Surveys and Tutorials*, 6(3):34 – 57, October 2004.
- [12] Scott Contini, Arjen K. Lenstra, and Ron Steinfeld. VSH: an efficient and provable collision resistant hash collision. In *Advances in Cryptology - Eurocrypt'06*, volume 4004 of *Lecture Notes in Computer Science*, pages 165 – 182, Saint Petersburg, Russia, May 2006. Springer - Verlag.
- [13] Amir F. Dana, Radhika Gowaikar, Ravi Palanki, Babak Hassibi, and Michelle Effros. Capacity of wireless erasure networks. *IEEE Transactions on Information Theory*, 52(3):789 – 804, March 2006.
- [14] Ufuk Demir and Ozlem Aktaş. Raptor versus Reed Solomon forward error correction codes. In *ISCN'06*, pages 264 – 269, Istanbul, Turkey, June 2006. IEEE.
- [15] Omid Etesami, Mehdi Molkarai, and Amin Shokrollahi. Raptor codes on symmetric channels. Available online at: <http://www.cs.berkeley.edu/~etesami/raptor.pdf>, preprint 2003.
- [16] Omid Etesami and Amin Shokrollahi. Raptor codes on binary memoryless symmetric channels. *IEEE Transactions on Information Theory*, 52(5):2033 – 2051, May 2006.
- [17] James Chuan Fu and W. Y. Wendy Lou. *Distribution Theory of Runs and Patterns and its Applications*. World Scientific Publishing, 2003.
- [18] Chongzhi Gao and Zhengnan Yao. How to authenticate real time streams using improved online/offline signatures. In *4th International Conference on Cryptology and Network Security*, volume 3810 of *Lecture Notes in Computer Science*, pages 134 – 146, Xiamen, China, December 2005. Springer-Verlag.
- [19] Rosario Gennaro and Pankaj Rohatgi. How to sign digital streams. In *Advances in Cryptology - Crypto'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 180–197, Santa Barbara, USA, August 1997. Springer-Verlag.
- [20] Phillippe Golle and Nagendra Modadugu. Authenticating streamed data in the presence of random packet loss. In *Network and Distributed Systems Security Symposium on*, pages 13 – 22, San Diego, USA, February 2001. Internet Society.

- [21] Venkatesan Guruswami. *List Decoding of Error-Correcting Codes*. Springer-Verlag, 2004.
- [22] Venkatesan Guruswami and Piotr Indyk. Linear-time decoding in error-free settings (extended abstract). In *ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 695 – 707, Turku, Finland, July 2004. Springer - Verlag.
- [23] Venkatesan Guruswami and Atri Rudra. Explicit capacity-achieving list-decodable codes. Technical Report TR05-133, Electronic Colloquium on Computational Complexity, November 2005.
- [24] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45(6):1757 – 1767, September 1999.
- [25] Chris Harrelson, Lawrence Ip, and Wei Wang. Limited randomness LT codes. In *41st Annual Allerton Conference on Communication, Control and Computing*, Urbana-Champaign, USA, October 2003.
- [26] Piotr Indyk. List-decoding in linear time. Technical Report TR02-024, Electronic Colloquium on Computational Complexity, April 2002.
- [27] Chris Karlof, Naveen Sastry, Yaping Li, Adrian Perrig, and J. D. Tygar. Distillation codes and applications to DoS resistant multicast authentication. In *11th Network and Distributed Systems Security Symposium (NDSS)*, San Diego, USA, February 2004.
- [28] Richard Karp, Michael Luby, and Amin Shokrollahi. Finite length analysis of LT codes. In *International Symposium on Information Theory*, page 39, Chicago, USA, June 2004. IEEE Press.
- [29] Richard Karp, Michael Luby, and Amin Shokrollahi. Verification decoding of Raptor codes. In *ISIT 2005*, pages 1310 – 1314, Adelaide, Australia, September 2005. IEEE.
- [30] Jérôme Lacan and Jérôme Fimes. Systematic MDS erasure codes based on Vandermonde matrices. *IEEE Communications Letters*, 8(9):570 – 572, September 2004.
- [31] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and their Applications - Revised Edition*. Cambridge University Press, 2000.
- [32] Michael Luby. LT codes. In *43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'02)*, pages 271 – 282, Vancouver, Canada, November 2002. IEEE Computer Society.
- [33] Michael Luby, Mark Watson, Tiago Gasiba, Thomas Stockhammer, and Wen Xu. Raptor codes for reliable download delivery in wireless broadcast systems. In *CCNC 2006*, pages 192 – 197, Las Vegas, USA, January 2006. IEEE Press.
- [34] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, and Daniel A. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569 – 584, February 2001.
- [35] Anna Lysyanskaya, Roberto Tamassia, and Nikos Triandopoulos. Multicast authentication in fully adversarial networks. In *IEEE Symposium on Security and Privacy*, pages 241 – 253, Oakland, USA, May 2003. IEEE Press.
- [36] Florence Jessiem MacWilliams and Neil James Alexander Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.
- [37] Petar Maymoukov. Online codes. Technical report, New York University, November 2002.
- [38] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
- [39] Ralph Merkle. A certified digital signature. In *Advances in Cryptology - Crypto'89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238, Santa Barbara, USA, August 1989. Springer - Verlag.
- [40] Sarah Miner and Jessica Staddon. Graph-based authentication of digital streams. In *IEEE Symposium on Security and Privacy*, pages 232 – 246, Oakland, USA, May 2001. IEEE Press.
- [41] National Institute of Standards and Technology. FIPS 180-2: Secure Hash Standard (SHS). Available online at: <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf>, August 2002. Amended 25 February 2004.
- [42] Lan Nguyen. Accumulators from bilinear pairings and applications. In *Topics in Cryptology CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 275 – 292, San Francisco, USA, February 2005. Springer - Verlag.
- [43] Kaisa Nyberg. Fast accumulated hashing. In *Fast Software Encryption - Third International Workshop*, volume 1039 of *Lecture Notes in Computer Science*, pages 83 – 87, Cambridge, United Kingdom, February 1996. Springer.

- [44] Ravi Palanki and Jonathan S. Yedidia. Rateless codes on noisy channels. In *38th Annual Conference on Information Sciences and Systems*, Princeton, USA, March 2004.
- [45] Alain Pannetrat and Rafik Molva. Authenticating real time packet streams and multicasts. In *7th International Symposium on Computers and Communications*, Taormina, Italy, July 2002. IEEE Computer Society.
- [46] Jung Min Park, Edwin K. P. Chong, and Howard Jay Siegel. Efficient multicast packet authentication using signature amortization. In *IEEE Symposium on Security and Privacy*, pages 227 – 240, Oakland, USA, May 2002. IEEE Press.
- [47] Jung Min Park, Edwin K. P. Chong, and Howard Jay Siegel. Efficient multicast stream authentication using erasure codes. *ACM Transactions on Information and System Security*, 6(2):258 – 285, May 2003.
- [48] Yongsu Park and Yookun Cho. The eSAIDA stream authentication scheme. In *ICCSA*, volume 3046 of *Lecture Notes in Computer Science*, pages 799 – 807, San Diego, USA, April 2004. Springer - Verlag.
- [49] Vern Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277 – 292, June 1999.
- [50] Adrian Perrig, Ran Canetti, J.D. Tygar, and Dawn Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56 – 73, Oakland, USA, May 2000. IEEE Press.
- [51] Adrian Perrig and J. D. Tygar. *Secure Broadcast Communication in Wired and Wireless Networks*. Kluwer Academic Publishers, 2003.
- [52] Josef Pieprzyk, Thomas Hardjono, and Jennifer Seberry. *Fundamentals of Computer Security*. Springer, 2003.
- [53] Hossein Pishro-Nik and Faramarz Fekri. On Raptor codes. In *2006 IEEE International Conference on Communications*, pages 1137 – 1141, Istanbul, Turkey, June 2006. IEEE.
- [54] Malempati Madhusudana Rao. *Conditional Measures and Applications (Second Edition)*. CRC Press, 2005.
- [55] Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of Society for Industrial and Applied Mathematics*, 8(2):300 – 304, June 1960.
- [56] Ronald L. Rivest, Adi Shamir, and Len Adelman. A method for obtaining digital signatures and public key cryptosystems. *Communication of the ACM*, 21(2):120 – 126, February 1978.
- [57] Pankaj Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *6th ACM Conference on Computer and Communications Security*, pages 93 – 100, Singapore, November 1999. ACM Press.
- [58] Ron M. Roth and Vitaly Skachek. Improved nearly-MDS expander codes. Available online at: http://arxiv.org/PS_cache/cs/pdf/0601/0601090.pdf, January 2005.
- [59] Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In *Advances in Cryptology - Crypto'01*, volume 2139 of *Lecture Notes in Computer Science*, pages 355 – 367, Santa Barbara, USA, August 2001. Springer - Verlag.
- [60] Amin Shokrollahi. Raptor codes. *IEEE Transactions on Information Theory*, 52(6):2551 – 2567, June 2006.
- [61] Douglas R. Stinson. *Cryptography: Theory and Practice (Third Edition)*. Chapman & Hall/CRC, 2006.
- [62] Christophe Tartary and Huaxiong Wang. Efficient multicast stream authentication for the fully adversarial network. *International Journal of Security and Network (Special Issue on Cryptography in Networks)*, to appear. Inderscience.
- [63] Christophe Tartary and Huaxiong Wang. Efficient multicast stream authentication for the fully adversarial network. In *6th International Workshop on Information Security Applications*, volume 3786 of *Lecture Notes in Computer Science*, pages 108 – 125, Jeju Island, Korea, August 2005. Springer - Verlag.
- [64] Christophe Tartary and Huaxiong Wang. Achieving multicast stream authentication using MDS codes. In *5th International Conference on Cryptology and Network Security*, volume 4301 of *Lecture Notes in Computer Science*, pages 108 – 125, Suzhou, China, December 2006. Springer - Verlag.
- [65] Christophe Tartary and Huaxiong Wang. Rateless codes for the multicast stream authentication problem. In *1st International Workshop on Security*, volume 4266 of *Lecture Notes in Computer Science*, pages 136 – 151, Kyoto, Japan, October 2006. Springer - Verlag.
- [66] Dejan Vukobratovic and Miroslav Despotovic. On the packet lengths of rateless codes. In *EUROCON 2005*, pages 672 – 675, Belgrade, Serbia & Montenegro, November 2005. IEEE.

- [67] Chung Kei Wong and Simon S. Lam. Digital signatures for flows and multicasts. *IEEE/ACM Transactions on Networking*, 7(4):502 – 513, August 1999.
- [68] Qian Xu, Vladimirm Stanković, and Zixiang Xiong. Distributed joint source-channel coding of video using Raptor codes. In *DCC 2005*, page 491, Snowbird, USA, March 2005. IEEE Computer Society.
- [69] Maya Yajnik, Sue Moon, Jim Kurose, and Don Towsley. Measurement and modeling of the temporal dependence in packet loss. In *IEEE Conference on Computer Communications*, volume 1, pages 345 – 352, New York, USA, March 1999. IEEE Press.
- [70] Jean-Pierre Zanotti. Le code correcteur C.I.R.C. Available online at: <http://zanotti.univ-tln.fr/enseignement/divers/chapter3.html>.

A Asymptotic Behavior of the Ratio $\frac{\mathcal{N}}{n}$

In this appendix, we demonstrate that the ratio for our rateless code construction from Section 4.1 turns out to be $O(1)$ as a function of the block length n .

Due to the definition of \mathcal{N} , we have:

$$\frac{\mathcal{N}}{n} \leq 1 + \frac{1}{\alpha} \frac{m}{n}$$

Thus, it is sufficient to prove that $\frac{m}{n}$ is $O(1)$ as a function of n . We have:

$$\frac{m}{n} = 1 + \frac{1}{\sqrt{n}} \left(c \ln \left(\frac{n}{\delta} \right) \ln \left(\frac{c \ln \left(\frac{n}{\delta} \right) \sqrt{n}}{\delta} \right) \right) \left(\sum_{i=1}^{n-R} \frac{1}{i} \right)$$

We can find an upper bound on the sum as follows:

$$\sum_{i=1}^{n-R} \frac{1}{i} \leq \sum_{i=1}^n \frac{1}{i} \leq 1 + \int_2^n \frac{1}{x} dx \leq 1 - \ln 2 + \ln n$$

Therefore, when n is large, we get:

$$\frac{m}{n} \leq 1 + 2c \frac{\ln n \left(\ln \left(\frac{n}{\delta} \right) \right)^2}{\sqrt{n}} \leq 1 + 2c \frac{\left(\ln \left(\frac{n}{\delta} \right) \right)^3}{\sqrt{n}}$$

Due to our choice of c in Section 2.2, we get:

$$\frac{m}{n} \leq 1 + \frac{\left(\ln \left(\frac{n}{\delta} \right) \right)^2}{n^{\frac{1}{2}-\eta}}$$

As $\eta < \frac{1}{2}$ and δ is a constant, we obtain our result.

B Proof of Theorem 3

Assume that the scheme is either insecure or not (α, β) -correct. By definition, a probabilistic polynomial-time opponent \mathcal{O} can break the scheme security or correctness with a non-negligible probability $\pi(k)$ where k is the security parameter setting up the digital signature and the hash function. As a probability is a measure [4, 54], we must have either cases:

- (1) With probability at least $\pi(k)/2$, \mathcal{O} breaks the scheme correctness.
- (2) With probability at least $\pi(k)/2$, \mathcal{O} breaks the scheme security.

It should be noticed that since $\pi(k)$ is a non-negligible function of k , so is $\pi(k)/2$.

Point (1). We will demonstrate by contradiction that if \mathcal{O} can break the scheme correctness in polynomial time then either he can forge the digital signature or he can find a collision for the hash function in polynomial time as well.

This will be proved by turning an attack breaking the (α, β) -correctness of our construction into a successful attack against either the digital signature or the hash function.

For this attack, \mathcal{O} will have access to the signing algorithm Sign_{SK} (but \mathcal{O} will not have access to SK itself). He can use the public key PK as well as the collision resistant hash function h . \mathcal{O} will be allowed to run Authenticator1 whose queries are written as $(\text{BID}_i, n_i, \alpha_i, \beta_i, \rho_i, \delta_i, \text{DP}_i)$ where DP_i is the set of n_i data packets to be authenticated. In order to get the corresponding output, the signature is obtained by querying Sign_{SK} as a black-box at Step 2 of Authenticator1.

According to our hypothesis, \mathcal{O} broke the correctness of the construction. This means that, following the previous process, \mathcal{O} managed to obtain values $\text{BID}, n, \alpha, \beta, \rho, \delta$ and a set of received packets RP such that:

- $\exists i : (\text{BID}, n, \alpha, \beta, \rho, \delta) = (\text{BID}_i, n_i, \alpha_i, \beta_i, \rho_i, \delta_i)$.
Denote $\text{DP} = \{P_1, \dots, P_n\} (= \text{DP}_i)$ the n data packets associated with this query and AP the response given to \mathcal{O} . In particular, we denote σ the signature corresponding to DP and generated as in Step 2 of Authenticator1.
- $|\text{RP} \cap \text{AP}| \geq \lceil \alpha \mathcal{N} \rceil$ and $|\text{RP}| \leq \lfloor \beta \mathcal{N} \rfloor$.
- $\{P'_1, \dots, P'_n\} = \text{Decoder1}(\text{PK}, \text{BID}, n, \alpha, \beta, \rho, \delta, \text{List}, \text{RP})$ where $P'_j \notin \{\emptyset, P_j\}$ for some $j \in \{1, \dots, n\}$.

In the previous statement, List designates the list of irreducible polynomials from Table 1. As said in Section 4.1, it is assumed that List contains only one polynomial $\mathcal{Q}(X)$ per degree value and the elements \mathcal{N} and q as well as the pad length ℓ can be determined from the content of that table.

Assume that the digital signature is unforgeable and the hash function is collision resistant.

Since $|\text{RP} \cap \text{AP}| \geq \lceil \alpha \mathcal{N} \rceil$ and $|\text{RP}| \leq \lfloor \beta \mathcal{N} \rfloor$, Step 1 of DecoderLTscheme ends successfully. The consistency of Poly-Reconstruct involves that the list returned by MPR at Step 2 contains the element $h_1 \| \dots \| h_{\mathcal{N}} \| \sigma$ corresponding to DP after removing the pad of length ℓ .

As the digital signature is unforgeable and the hash function is collision resistant, the pair message/signature going through the verification process at Step 3 correspond to DP. Therefore, at the end of that step, we have:

$$\forall i \in \{1, \dots, \mathcal{N}\} \quad h'_i = h(E_i \| N_i^1 \| \dots \| N_i^{d_i})$$

For the same reason as before, at the end of Step 4, either $E'_i = \emptyset$ or we have:

$$E'_i = E_i \text{ and } d'_i = d_i \text{ and } \forall \xi \in \{1, \dots, d_i\} \quad N_i'^{\xi} = N_i^{\xi}$$

There are at least $\lceil \alpha \mathcal{N} \rceil \leq m$ non-empty elements. So, at Step 5, either the LT decoder output the n original packets P_1, \dots, P_n (which happens with probability at least $1 - \delta$) or output χ elements along with $n - \chi$ empty symbols. Nevertheless, when looking at the LT decoding process [32], we notice that the χ non-empty elements must belong to $\{P_1, \dots, P_n\}$ since they represent the data packets which were released from the ripple when it vanished (in other words, the decoding process of LT codes is consistent). Therefore, we get:

$$\forall i \in \{1, \dots, n\} \quad P'_i \in \{\emptyset, P_i\}$$

We obtain a contradiction with our original hypothesis which stipulated:

$$\exists j \in \{1, \dots, n\} \quad P'_j \notin \{\emptyset, P_j\}$$

As a consequence, we deduce that either the hash function is not collision resistant or the digital signature is not secure.

Point (2). We will demonstrate by contradiction that if \mathcal{O} can break the scheme correctness in polynomial time then either he can forge the digital signature or he can find a collision for the hash function in polynomial time as well.

We consider the same kind of reduction as in Point (1). The opponent \mathcal{O} breaks the security of the scheme if one of the following holds:

- I. Authenticator1 was never queried on input $\text{BID}, n, \alpha, \beta, \rho, \delta$ and the decoding algorithm Decoder1 does not reject RP, i.e. $\{P'_1, \dots, P'_n\} \neq \emptyset$ where:
 $\{P'_1, \dots, P'_n\} = \text{Decoder1}(\text{PK}, \text{BID}, n, \alpha, \beta, \rho, \delta, \text{List}, \text{RP})$.
- II. Authenticator1 was queried on input $\text{BID}, n, \alpha, \beta, \rho, \delta$ for some data packets $\text{DP} = \{P_1, \dots, P_n\}$. Nevertheless, the output of Decoder1 verifies $P'_j \notin \{\emptyset, P_j\}$ for some $j \in \{1, \dots, n\}$.

Case I. Since Decoder1 output some non-empty packets, Step 3 had to terminate successfully. Thus, it has been found a pair $(h(\text{BID} \| n \| \alpha \| \beta \| \delta \| h_1 \| \dots \| h_{\mathcal{N}}), \sigma)$ such that:

$$\text{Verify}_{\text{PK}}(h(\text{BID} \| n \| \alpha \| \beta \| \delta \| h_1 \| \dots \| h_{\mathcal{N}}), \sigma) = \text{TRUE}$$

If \mathcal{O} never queried Authenticator1 for block tag BID then the previous pair is a forgery of the digital signature.

If \mathcal{O} queried Authenticator1 for block tag BID then denote $(\text{BID}, \hat{n}, \hat{\alpha}, \hat{\beta}, \hat{\rho}, \hat{\delta})$ his query. By hypothesis, we have:

$$(\text{BID}, \hat{n}, \hat{\alpha}, \hat{\beta}, \hat{\rho}, \hat{\delta}) \neq (\text{BID}, n, \alpha, \beta, \rho, \delta)$$

As ρ is uniquely determined when n, α and β are given, we get:

$$(\text{BID}, \hat{n}, \hat{\alpha}, \hat{\beta}, \hat{\delta}) \neq (\text{BID}, n, \alpha, \beta, \delta)$$

Therefore, the previous pair is a forgery of the signature scheme.

Case II. We have the same situation as Point (1).

C Proof of Theorem 4 and Theorem 6

We decompose this proof into three parts. In the first one we will demonstrate the recovery property of our two schemes. In the second one, we will prove that $U(\mathcal{N})$ and $U(n)$ are upper bounds on the number of signature verifications to be performed per block for our two constructions. Finally, we will show that those bounds are $O(1)$ as functions of the block length n . Since our results have to be valid for any value BID, we start our proof by picking a value BID which will remain fixed throughout this proof.

C.1 Packet Recovery

We first focus on the LT code construction as the proof for the MDS-based scheme will be analogous. By definition of the rates, at least $\lceil \alpha \mathcal{N} \rceil$ of the original elements $\{\text{AP}_1, \dots, \text{AP}_{\mathcal{N}}\}$ are received by the receiver amongst a total of no-more than $\lfloor \beta \mathcal{N} \rfloor$ elements. Thus, the demonstration of Theorem 3 shows that Decoder1 returns P_1, \dots, P_n with probability at least $1 - \delta$ since the digital signature is unforgeable and the hash function is collision resistant.

The difference for our MDS code-based construction is that, at Step 6, Decoder2 recovers all n packets P_1, \dots, P_n with probability 1 as decoding performed at Step 5 is deterministic.

C.2 Signature Verifications

Since at most one signature verification is performed per element of the list output by MPR, it is sufficient to prove that $U(\mathcal{N})$ and $U(n)$ are upper bounds on the size of that list for our respective schemes.

As in the previous section, we first focus on the LT code-based protocol. Denote N the number of points on which Poly-Reconstruct is run and T the number of original elements in this list. By definition of α and β we have: $T \geq \lceil \alpha \mathcal{N} \rceil$ and $N \leq \lfloor \beta \mathcal{N} \rfloor$. As noticed before we have: $T > \sqrt{(\rho \mathcal{N}) N}$ which guarantees Poly-Reconstruct to be run successfully. Denote $L(N, T)$ the size of the list output by Poly-Reconstruct. We want to prove: $L(T, N) \leq U(\mathcal{N})$.

According to the proof of Proposition 6.15 in Guruswami's thesis [21], we have: $L(T, N) \leq \lfloor \frac{\ell}{k} \rfloor$ where:

$$\begin{cases} \ell = rT - 1 \\ r = 1 + \left\lfloor \frac{kN + \sqrt{k^2 N^2 + 4(T^2 - kN)}}{2(T^2 - kN)} \right\rfloor \end{cases}$$

In our case $k = \rho \mathcal{N}$. Therefore, an upper on $L(T, N)$ is given by:

$$\frac{T}{\rho \mathcal{N}} \left(1 + \frac{(\rho \mathcal{N}) N + \sqrt{(\rho \mathcal{N})^2 N^2 + 4(T^2 - (\rho \mathcal{N}) N)}}{2(T^2 - (\rho \mathcal{N}) N)} \right) - \frac{1}{\rho \mathcal{N}} \quad (3)$$

1st bound: We have: $\forall (a, b) \in \mathbb{R}^+ \times \mathbb{R}^+ \sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$. We deduce that $L(T, N)$ is upper bounded by:

$$\frac{T}{\rho \mathcal{N}} \left(1 + \frac{(\rho \mathcal{N}) N}{T^2 - (\rho \mathcal{N}) N} + \frac{1}{\sqrt{T^2 - (\rho \mathcal{N}) N}} \right) - \frac{1}{\rho \mathcal{N}}$$

Using $T \geq \lceil \alpha \mathcal{N} \rceil$, we deduce that $T^2 - (\rho \mathcal{N}) N$ is lower bounded by $\mathcal{N}^2 (\alpha^2 - \beta \rho)$. This element is positive since $\rho < \frac{\alpha^2}{\beta}$. Thus:

$$L(T, N) \leq \frac{T}{\rho \mathcal{N}} \left(1 + \frac{\rho \mathcal{N}}{\mathcal{N} (\alpha^2 - \beta \rho)} + \frac{1}{\mathcal{N} \sqrt{\alpha^2 - \beta \rho}} \right) - \frac{1}{\rho \mathcal{N}}$$

Since $T \leq \mathcal{N}$ and $N \leq \lfloor \beta \mathcal{N} \rfloor$, $U_1(\mathcal{N})$ is an upper bound of the right hand side of the previous inequality. Since $L(T, N)$ is an integer we get: $L(T, N) \leq \lfloor U_1(\mathcal{N}) \rfloor$.

2nd bound: We start again from (3). Since $T \leq \mathcal{N}$ we have: $\frac{T}{\rho \mathcal{N}} \leq \frac{1}{\rho}$. The numerator of the fraction is upper bounded by $\beta \rho \mathcal{N} + \sqrt{(\beta \rho \mathcal{N}^2)^2 + 4(\mathcal{N}^2 - \rho \alpha \mathcal{N}^2)}$. As before $T^2 - (\rho \mathcal{N}) N$ is lower bounded by $\mathcal{N}^2 (\alpha^2 - \beta \rho)$. Therefore:

$$L(T, N) \leq \frac{1}{\rho} \left(1 + \frac{\beta \rho + \sqrt{(\beta \rho)^2 + \frac{4}{\mathcal{N}^2} (1 - \rho \alpha)}}{2(\alpha^2 - \beta \rho)} \right) - \frac{1}{\rho \mathcal{N}}$$

The right hand side of the previous inequality is equal to $U_2(\mathcal{N})$. Therefore we have: $L(T, N) \leq \lfloor U_2(\mathcal{N}) \rfloor$.

Finally, we obtain: $L(T, N) \leq \min(\lfloor U_1(\mathcal{N}) \rfloor, \lfloor U_2(\mathcal{N}) \rfloor)$ which means $L(T, N) \leq U(\mathcal{N})$.

Concerning our MDS code-based scheme, it is easy to see that $U(n)$ is an upper bound on $L(T, N)$ in that case by substituting \mathcal{N} by n in the above proof.

C.3 Asymptotic Analysis

In this section, we start studying the MDS code construction. As in [35], we consider that ρ is a constant when studying the asymptotic behavior of $U(n)$. Nevertheless ρn must be an integer. Therefore the limit of $U(n)$ can only be studied for values n in $\mathcal{I} := \{n/\rho n \in \mathbb{N}\}$. A necessary and sufficient condition to study the limit in $+\infty$ is to have an infinite number of elements in \mathcal{I} since \mathcal{I} is a subset of \mathbb{N} . Remember that ρ is a (positive) rational number. Thus we can write $\rho = \frac{u_\rho}{v_\rho}$ where u_ρ and v_ρ are elements of \mathbb{N} . If we consider $\mathbb{N}v_\rho$, the subset of \mathbb{N} representing the multiples of v_ρ , then: $\mathbb{N}v_\rho \subset \mathcal{I}$. Since $\mathbb{N}v_\rho$ is infinite, so is \mathcal{I} . Therefore, we can study the asymptotic behavior of $U(n)$ as soon as ρ is a rational number. We have:

$$\begin{aligned} \lim_{\substack{n \rightarrow +\infty \\ n \in \mathcal{I}}} \left(\frac{1}{\rho n} \right) &= 0 \\ \lim_{\substack{n \rightarrow +\infty \\ n \in \mathcal{I}}} \left(\sqrt{\beta^2 + \frac{4}{\rho^2 n^2} (1 - \rho \alpha)} \right) &= \beta \end{aligned}$$

These two equations involve that $U_2(n)$ has a finite limit when n tends to $+\infty$ (and $n \in \mathcal{I}$). Thus we get $U_2(n) \in O(1)$ and then $\lfloor U_2(n) \rfloor \in O(1)$.

The left hand side equality involves that $U_1(n)$ has a finite limit when n tends to $+\infty$ (and $n \in \mathcal{I}$). As before we obtain: $\lfloor U_1(n) \rfloor \in O(1)$.

Since $U(n) = \min(\lfloor U_1(n) \rfloor, \lfloor U_2(n) \rfloor)$, we finally deduce: $U(n) \in O(1)$ as a function of n .

Concerning the LT code-based scheme, we could similarly demonstrate that $U(\mathcal{N}) \in O(1)$ as a function of \mathcal{N} . By construction we have $\mathcal{N} \geq n$. In Appendix A, we showed that the ratio $\frac{\mathcal{N}}{n}$ was $O(1)$ as a function of n . This involves that $\mathcal{N} \in O(n)$. Therefore, \mathcal{N} and n have the same order as functions of n (i.e. $\mathcal{N} \in \Theta(n)$). As a consequence, $U(\mathcal{N}) \in O(1)$ as a function of n as well.

D Pad Design

In this appendix, we detail the padding occurring for our two authentication schemes. We recall that \mathcal{P} denotes the bit size of any packet P_i , \mathcal{H} the bit size of digests produced by h and \mathcal{S} the bit size of signatures.

D.1 LT Code-based Authentication Scheme

We describe the length of the pad occurring at Step 3 of Algorithm 4 as well as the size of the corresponding field \mathbb{F}_{2^q} for polynomial evaluation.

At Step 3, one has to represent $\tau := h_1 \parallel \dots \parallel h_{\mathcal{N}} \parallel \sigma$ over $\rho \mathcal{N} + 1$ field elements. So, we have:

$$q \geq \left\lceil \frac{\mathcal{H} \mathcal{N} + \mathcal{S}}{\rho \mathcal{N} + 1} \right\rceil$$

It should be noticed that $A(X)$ is evaluated at \mathcal{N} distinct points of \mathbb{F}_{2^q} . Thus, we must have:

$$q \geq \lceil \log_2 \mathcal{N} \rceil$$

Therefore, we need to have:

$$\left\lceil \frac{\mathcal{H}\mathcal{N} + \mathcal{S}}{\rho\mathcal{N} + 1} \right\rceil \geq \lceil \log_2 \mathcal{N} \rceil \quad (4)$$

The previous equation represents a constructibility requirement for our scheme. Even if Inequality (4) is verified in most practical situations, we need to ensure the constructibility of our approach for any case. The solution to this problem is simple. Indeed, one needs to pre-pad τ with ξ zeros as $\tau \parallel 0^\xi$ where ξ is the smallest element in \mathbb{N} such that:

$$\left\lceil \frac{\mathcal{H}\mathcal{N} + \mathcal{S} + \xi}{\rho\mathcal{N} + 1} \right\rceil \geq \lceil \log_2 \mathcal{N} \rceil \quad (5)$$

Note that Inequality (4) corresponds to the case $\xi = 0$ in Inequality (5). Notice that Inequality (5) is the same as Inequality (1) appearing at Step 3 of Algorithm 4. Denote $b := \mathcal{H}\mathcal{N} + \mathcal{S} + \xi \bmod (\rho\mathcal{N} + 1)$. Thus, the length of the pad to be appended to τ at Step 3 is ℓ bits where:

$$\ell := \begin{cases} 0 & \text{if } b = 0 \\ \rho\mathcal{N} + 1 - b & \text{otherwise} \end{cases}$$

So, we get:

$$q = \left\lceil \frac{\mathcal{H}\mathcal{N} + \mathcal{S} + \xi}{\rho\mathcal{N} + 1} \right\rceil$$

Important Remark. We can notice that information stored in the public Table 1 is sufficient to compute of the previous pad of length ℓ and q .

D.2 MDS Code-based Authentication Scheme

We describe the length of the pad occurring at Step 3 of Algorithm 6 as well as the size of the corresponding field \mathbb{F}_{2^q} for polynomial evaluation.

At Step 3, one has to represent $\tau := h_1 \parallel \dots \parallel h_n \parallel \sigma$ over $\rho n + 1$ field elements and perform n evaluations of the polynomial $A(X)$. Applying the same reasoning as in Appendix D.1, we need to pre-pad τ with ξ zeros as $\tau \parallel 0^\xi$ where ξ is the smallest element in \mathbb{N} such that:

$$\left\lceil \frac{\mathcal{H}n + \mathcal{S} + \xi}{\rho n + 1} \right\rceil \geq \lceil \log_2 n \rceil$$

The above inequality is the same as Inequality (2) appearing at Step 3 of Algorithm 6. Denote $b := \mathcal{H}n + \mathcal{S} + \xi \bmod (\rho n + 1)$. Thus, the length of the pad to be appended to τ at Step 3 is ℓ bits where:

$$\ell := \begin{cases} 0 & \text{if } b = 0 \\ \rho n + 1 - b & \text{otherwise} \end{cases}$$

So, we get:

$$q = \left\lceil \frac{\mathcal{H}n + \mathcal{S} + \xi}{\rho n + 1} \right\rceil$$

As for the LT code-based authentication scheme, the information stored in the public Table 3 is sufficient to compute of the previous pad of length ℓ and q .