

Rateless Codes for the Multicast Stream Authentication Problem*

Christophe Tartary and Huaxiong Wang

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

{ctartary, hwang}@ics.mq.edu.au

Abstract

We study the multicast authentication problem when an opponent can drop, reorder and introduce data packets into the communication channel. We first study the packet authentication probability of a scheme proposed by Lysyanskaya, Tamassia and Triandopoulos in 2003 since our opponent model is based on theirs. Using a family of rateless codes called Luby Transform codes (LT codes) we design a protocol which allows any packet to be authenticated at the receiver with probability arbitrary close to 1. We also compare LT codes to other families of rateless codes which could be used in that context in order to minimize the packet overhead as well as the time complexity of encoding and decoding data.

Keywords: Stream Authentication, Polynomial Reconstruction, Rateless 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. Unconditionally secure schemes have been designed in [1, 4, 25] but either they can only be used for a single authentication or they require too large storage capacities. In this paper, we will consider that opponents have bounded computational abilities.

In recent years, several protocols were designed to deal with the multicast authentication problem [3]. Applications like stock quotes and pay-TV involve that the stream size can be large and eventually infinite. On the other hand, the receivers must be able to authenticate data within a short period of delay upon reception. Since many protocols will transfer private or sensitive information, non-repudiation of the sender is required for most of them. Signing each packet¹ via digital signatures is impractical since they are generally time expensive to generate and verify. Using one-time or k -time signatures [6, 24] overcomes this drawback but their size is, in general, too large to be used for broadcasting due to bandwidth limitations. Thus, to provide non-repudiation, most techniques rely on signature amortization. One signature is created and its overhead and time generation/verification are amortized over several packets using hash functions.

In order to deal with packet loss, Perrig *et al.* designed EMSS [21] and MESS [21] where the hash of each packet is appended to several followers according to a specific pattern. One signature is generated from time to time and is always assumed to be received providing non-repudiation and allowing new users to join the communication group at any time. Using k -state Markov chains [20, 30] to model packet loss, they computed bounds on packet authentication probability. Using augmented chains to distribute hashes, Golle and Modadugu [7] and Miner and Staddon [17] obtained other bounds. Unfortunately, all these schemes rely on the reception of signed packets.

Wong and Lam [29] used Merkle-hash trees [16] to distribute hashes and solve the problem of reliable signature transmission. Their scheme can tolerate any packet loss and data injections but the size of the tag² is logarithmic in the number of packets per block³. As said earlier, bandwidth limitations prevent us from using such an approach.

*The original version of this paper appears in the proceedings of the 1st International Workshop on Security (IWSEC 2006), Lecture Notes in Computer Science, vol. 4266, pp 136 - 151, Springer - Verlag.

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

²We call *authentication tag* the extra information appended to a packet to prove its authenticity.

³In order to be processed, packets are gathered into fixed-size sets called *blocks*.

To deal with packet loss without relying on reception of the signature packets one can split the signature into k parts where only l of them ($l < k$) are enough for recovery. Using error-correcting codes, Lysyanskaya *et al.* [14] developed a protocol which also tolerates data injections. Their approach was recently extended by Tartary and Wang [28]. Nevertheless none of these schemes provided bounds on packet authentication probability. In addition the rate of their linear code depends on some network parameters. If one of them changes then the whole structure of the code must be updated.

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 [7, 17, 21, 29] or code symbols [14, 19, 28]. If \tilde{P} is dropped then P is definitely lost since such a \tilde{P} is unique for each scheme. As these previous techniques, we will process data stream packet per block of n elements: P_1, \dots, P_n . Our technique can be seen as an extension of Lysyanskaya *et al.*'s approach which enables any receiver to recover all data packets P_1, \dots, P_n despite loss incurred during transmission. This constitutes a major improvement from existing techniques 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 technique prevents frozen images and audio gaps to happen.

We propose to use **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 3). LT codes were introduced by Luby [12] as the first practical realization of rateless codes to illustrate the Digital Fountain concept [2]. 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 [8] and will recover that polynomial despite potential data injections performed by malicious users.

In [14], Reed-Solomon codes were used to design a multicast authentication scheme dealing with both packet loss and data injection. We will prove that its packet authentication probability (which has not been studied in [14]) does not exhibit an easy lower bound. In addition, when a packet was dropped during transmission its content was definitely lost for the receivers. Combining a LT code with the polynomial reconstruction technique we design a broadcast protocol having two main advantages. First, it allows the receivers to reconstruct erased data which is, to our knowledge, a new feature in the multicast authentication problem. Second, it exhibits a minimal lower bound on the packet authentication probability which can be chosen arbitrary close to 1.

Since we will use the same opponent model as Lysyanskaya *et al.*, we will prove that our scheme is as secure as theirs. Since its security depends on the consistency of the LT decoding (while Lysyanskaya *et al.*'s relies on Reed-Solomon codes' one), we will compare LT codes to other families of rateless codes including Online and Raptor codes [15, 26]. We will show that it is possible to achieve reasonable and constant 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. A digital signature will be used to ensure non-repudiation and to enable new members to join the communication group at any block boundary.

The paper is organized as follows. We will describe the scheme developed in [14] and analyze its packet authentication probability in the next section. In Section 3, we will develop our authentication protocol using LT codes. In Section 4, we will compare different families of rateless codes that could be used instead of LT codes. The last section will summarize our contribution to the multicast authentication problem.

2 Analysis of Lysyanskaya *et al.*'s Protocol

In this section, we will shortly describe the scheme designed in [14]. We will first introduce the network model used in that paper. Then, we will recall the tasks performed at the sender and the receiver to authenticate data and analyze the packet authentication probability of that approach since it has not been studied in the original paper [14]. Finally, we will illustrate our result when the network exhibit a bursty loss pattern [20].

2.1 Network Model

The network is assumed to be under partial control of an opponent \mathcal{O} who can drop and rearrange packets of his choice. He can also inject data into the channel. Since our main concern is the multicast authentication problem, we assume that a reasonable number of packets reaches the different receivers and not too many packets are injected by \mathcal{O} . Indeed, if too many packets are dropped then data transmission becomes the main problem to solve since the small number of received elements would be useless even authenticated. On the other hand, if \mathcal{O} can introduce a large quantity of packets then \mathcal{O} can potentially overflow the network and the major problem becomes strengthening the channel against denial-of-service

attacks.

The stream is split into blocks of n packets and we introduce two parameters: α ($0 < \alpha \leq 1$) (the *survival* rate) and β ($\beta \geq 1$) (the *flood* rate). It is assumed that at least αn original packets and a total of no more than βn packets reach each receiver.

2.2 Description of the Scheme

We need a collision-resistant hash function h [22] and a secure signature scheme $(\text{Sign}_{\text{SK}}, \text{Verify}_{\text{PK}})$ [27] the couple of keys of which (SK, PK) is created by a generator KeyGen . Denote $\{P_1, \dots, P_n\}$ the block of n packets to be sent. The number BID denotes the block position within the whole stream. Each P_i is hashed into h_i by h . A signature σ is generated as: $\sigma := \text{sign}_{\text{SK}}(\text{BID} \| h_1 \| \dots \| h_n)$. The concatenation $\mathcal{C} := h_1 \| \dots \| h_n \| \sigma$ is encoded into $[c_1, \dots, c_n]$ using the $[n, \rho n]_q$ binary systematic Reed-Solomon (SRS) code over \mathbb{F}_q where q is a power of 2. Both ρ and q depend on α and β . The sender builds the set of n authenticated packets $\{\text{AP}_1, \dots, \text{AP}_n\}$ to be emitted to the receivers via the network as: $\forall i \in \{1, \dots, n\} \text{AP}_i = \text{BID} \| i \| P_i \| c_i$.

When a receiver gets m elements $\{\text{AP}'_1, \dots, \text{AP}'_m\}$ he first decomposes each of them as: $\text{BID}_i \| j_i \| P'_{j_i} \| c'_{j_i}$. He verifies that $\text{BID}_i = \text{BID}$ and builds the family $\{(j_1, c'_{j_1}), \dots, (j_m, c'_{j_m})\}$. He checks that m is consistent with the values of the rates α and β . In order to deal with packet loss and data injections, he uses an algorithm developed by Guruswami and Sudan [8] (GS-Decoder) to list-decode the SRS code. He gets a list of candidates for the signature verification. If one of them verifies the signature then this element must be \mathcal{C} since the signature scheme is secure. Thus, the receiver recovers the hashes of the original packets. What remains to achieve is to authenticate the original packets amongst the received ones. The receiver computes the hashes of the m packets $P'_{j_1}, \dots, P'_{j_m}$ and look for matchings with the h_i 's. Since h is collision resistant if $h'_{j_k} = h_i$ then $P'_{j_k} = P_i$. In this case, P_i is said to have been *authenticated* by the receiver. Using this process he can find the original packets amongst data he received.

This authentication scheme deals with both packets loss and data injections and the communication group is joinable by new users at any block boundary. In addition no reliable transmission of the block signature is assumed. Nevertheless, no study of its packet authentication probability was performed in [14]. We will now address this point.

2.3 Analysis of the Packet Authentication Probability.

We call the action \mathcal{O} performs on the stream a *modification pattern*. We first introduce the following definition:

Definition 1 We say that a couple (α, β) of survival and flood rates is *accurate* for a network flow of n symbols if when data is sent per block of n elements, the receiver gets at least αn of them and receives no more than βn pieces of data (including opponent's injections). In addition, (α, β) is *unique* (and called the *accuracy* of the network) if the following two conditions occur:

1. \mathcal{O} can use a modification pattern \mathcal{M}' allowing to receive (at least) one set of αn original packets with positive probability as well as a modification pattern \mathcal{M}'' allowing to receive (at least) one set of βn packets with positive probability.
2. \mathcal{O} cannot use either a modification pattern allowing to receive less than αn original packets with a non-zero probability or a modification pattern allowing to receive more than βn packets with a non-zero probability.

The previous definition means that the accuracy (α, β) is optimal in the sense that no receiver can get less than αn original elements but \mathcal{O} can drop packets in such a way that at least one of the receivers can get exactly αn of them (pattern \mathcal{M}'). It also means that no receiver can get more than βn elements but \mathcal{O} can inject packets in such a way that at least one of the receivers can get exactly βn elements (pattern \mathcal{M}'').

From now on, we consider that (α, β) is the accuracy of the network. The set of n elements of Definition 1 will be $\{\text{AP}_1, \dots, \text{AP}_n\}$. Denote \mathcal{F} the set of families having at least αn elements of $\{\text{AP}_1, \dots, \text{AP}_n\}$. For any $\lambda \in \{\alpha n, \dots, n\}$ we also define \mathcal{F}_λ the subset of \mathcal{F} consisting of families having exactly λ elements. Denote \mathcal{M} the modification pattern used by \mathcal{O} . It induces a probabilistic distribution over \mathcal{F} and therefore over $\{P_1, \dots, P_n\}$. Our aim is to compute $\text{P}_{\mathcal{M}}(P_i \text{ is authenticated})$ for any $i \in \{1, \dots, n\}$. Assume that we received a family of packets F for the block number BID . We denote \hat{F} the subfamily of F consisting of the original packets. Since (α, β) is the accuracy of the network, we have: $\alpha n \leq |\hat{F}| \leq |F| \leq \beta n$. We have the following theorem:

Theorem 1 ([28]) *If (α, β) is the accuracy of the network (for a flow n) then any received family F verifies the signature.*

Therefore, we get for any i in $\{1, \dots, n\}$:

$$\mathbb{P}_{\mathcal{M}}(P_i \text{ is authenticated}) = \mathbb{P}_{\mathcal{M}}(P_i \text{ is authenticated} \mid F \text{ verifies the signature})$$

According to [14] once F verifies the signature, P_i is authenticated if and only if there exists a received element $\text{BID}\|k\|P'_k\|c'_k$ such that $h(P'_k) = h_i$. Given that h is collision resistant this happens if and only if $P'_k = P_i$. We have three cases. First, we have $\text{AP}_i = \text{BID}\|k\|P'_k\|c'_k$. Second, we have $\text{AP}_i \neq \text{BID}\|k\|P'_k\|c'_k$ but there is another original element $\text{AP}_j (j \neq i)$ such that $\text{AP}_j = \text{BID}\|k\|P'_k\|c'_k$ (this corresponds to the fact that a piece of data P_i has to be sent several times). Third, $\text{BID}\|k\|P'_k\|c'_k$ does come from the sender and therefore has been introduced by \mathcal{O} . Since \mathcal{O} can eavesdrop the network, he knows all the AP_j 's. Since he has no interest in helping the receivers to get original data, he will only introduce incorrect content. Thus, we can claim the last two cases have a very small probability to happen and approximate the right hand side of the inequality by $\mathbb{P}_{\mathcal{M}}(\text{AP}_i \text{ is received})$:

$$\mathbb{P}_{\mathcal{M}}(P_i \text{ is authenticated}) \simeq \mathbb{P}_{\mathcal{M}}(\text{AP}_i \text{ is received}) \quad (1)$$

Since AP_i is an original packet, we have: $\mathbb{P}_{\mathcal{M}}(\text{AP}_i \text{ is received}) = \mathbb{P}_{\mathcal{M}}(\text{AP}_i \in \hat{F})$. Furthermore, the cardinality of \hat{F} belongs to $\{\alpha n, \dots, n\}$. So, we can write:

$$\begin{aligned} \mathbb{P}_{\mathcal{M}}(\text{AP}_i \in \hat{F}) &= \mathbb{P}_{\mathcal{M}}\left(\bigcup_{\lambda=\alpha n}^n \left\{ \hat{F} \in \mathcal{F}_\lambda \text{ and } \text{AP}_i \in \hat{F} \right\}\right) \\ &= \sum_{\lambda=\alpha n}^n \mathbb{P}_{\mathcal{M}}(\hat{F} \in \mathcal{F}_\lambda \text{ and } \text{AP}_i \in \hat{F}) \end{aligned}$$

The last equality comes from the fact that $\{\mathcal{F}_{\alpha n}, \dots, \mathcal{F}_n\}$ is a partition of \mathcal{F} . The distribution induced by \mathcal{M} may involve $\mathbb{P}_{\mathcal{M}}(\hat{F} \in \mathcal{F}_\lambda) = 0$ for some values of λ . In this case $\mathbb{P}_{\mathcal{M}}(\text{AP}_i \in \hat{F} \mid \hat{F} \in \mathcal{F}_\lambda)$ may not be uniquely defined [23] but the product $\mathbb{P}_{\mathcal{M}}(\text{AP}_i \in \hat{F} \mid \hat{F} \in \mathcal{F}_\lambda) \mathbb{P}_{\mathcal{M}}(\hat{F} \in \mathcal{F}_\lambda)$ is still equal to 0. Thus, we get a unique value for $\mathbb{P}_{\mathcal{M}}(\text{AP}_i \in \hat{F})$ as:

$$\mathbb{P}_{\mathcal{M}}(\text{AP}_i \in \hat{F}) = \sum_{\lambda=\alpha n}^n \mathbb{P}_{\mathcal{M}}(\text{AP}_i \in \hat{F} \mid \hat{F} \in \mathcal{F}_\lambda) \mathbb{P}_{\mathcal{M}}(\hat{F} \in \mathcal{F}_\lambda) \quad (2)$$

By combining Approximation (1) and Equality (2), we get an approximation of the packet authentication probability of Lysyanskaya *et al.*'s scheme as:

$$\mathbb{P}_{\mathcal{M}}(P_i \text{ is authenticated}) \simeq \sum_{\lambda=\alpha n}^n \mathbb{P}_{\mathcal{M}}(\text{AP}_i \in \hat{F} \mid \hat{F} \in \mathcal{F}_\lambda) \mathbb{P}_{\mathcal{M}}(\hat{F} \in \mathcal{F}_\lambda) \quad (3)$$

Definition 1 tells us that there exists a pattern \mathcal{M}' such that $\mathbb{P}_{\mathcal{M}}(\hat{F} \in \mathcal{F}_{\alpha n}) \neq 0$ so that the starting index value αn cannot be increased in the general setting.

2.4 Resistance against Bursty Loss

In [20], Paxson outlined that the Internet experienced bursty packet loss. Golle and Modadugu [7] and Miner and Stadon [17] designed schemes based on graph theory to resist multiple bursts of fixed lengths. We will illustrate an application of Approximation (3) when \mathcal{M} is a multiple-burst pattern in order to approximate $\mathbb{P}_{\mathcal{M}}(P_i \text{ is authenticated})$. In this case, there are no packet injections so $\beta = 1$. Due to space limitations, we only give milestones to follow.

Definition 2 A **burst** is a sequence of consecutive deletions. Two bursts are separated by at least one non-erased element.

We must determine how many bursts can occur over $\{\text{AP}_1, \dots, \text{AP}_n\}$ providing that at least αn of these elements are received.

Definition 3 Let (t_1, \dots, t_n) be a n -tuple. We say that a burst of length $b (\leq n)$ **starts** (respectively **ends**) at t_i if the elements erased by the burst are t_i, \dots, t_{i+b-1} (respectively t_{i-b+1}, \dots, t_i).

Definition 4 A tuple of positive integers $(\mathcal{B}_1, p_1, \dots, \mathcal{B}_\delta, p_\delta)$ is called a δ -burst if for all $i \in \{1, \dots, \delta\}$, \mathcal{B}_i is the length of the i^{th} burst occurring over a tuple (t_1, \dots, t_n) and starts at position p_i . $(\mathcal{B}_1, \dots, \mathcal{B}_\delta)$ is called the δ -length of the δ -burst.

It can be shown that, in order to have a δ -burst over a set of n elements, we must have: $n \geq \mathcal{B}_1 + \dots + \mathcal{B}_\delta + (\delta - 1)$. Since at least αn original elements have to be received, δ must not exceed $\min(\lfloor \frac{n+1}{2} \rfloor, (1 - \alpha)n)$ either. Once δ and the δ -length $(\mathcal{B}_1, \dots, \mathcal{B}_\delta)$ are chosen, it can be proved that there are:

$$N = \sum_{i_\delta=1}^{\xi+1} \sum_{i_{\delta-1}=1}^{i_\delta} \dots \sum_{i_2=1}^{i_3} \sum_{i_1=1}^{i_2} 1$$

possibilities to choose the starting positions (p_1, \dots, p_δ) where ξ is the unique natural integer such that: $n = \mathcal{B}_1 + \dots + \mathcal{B}_\delta + (\delta - 1) + \xi$. This value N represents the number of ways one can build a δ -burst $(\mathcal{B}_1, p_1, \dots, \mathcal{B}_\delta, p_\delta)$ over $\{\text{AP}_1, \dots, \text{AP}_n\}$. Since we want AP_i to be received, we must determine the number N_i of such δ -bursts which do not erase AP_i . We can assume that \mathcal{O} chooses any δ -burst with equal likelihood. Thus:

$$P_{\mathcal{M}}(\text{AP}_i \text{ is received}) = \frac{N_i}{N}$$

If we denote k_i (respectively k'_i) the maximal number of bursts which can occur over $\{\text{AP}_1, \dots, \text{AP}_{i-1}\}$ (respectively $\{\text{AP}_{i+1}, \dots, \text{AP}_n\}$) then we have to compute how many Δ -bursts $(\mathcal{B}_1, p_1, \dots, \mathcal{B}_\Delta, p_\Delta)$ can occur over $\{\text{AP}_1, \dots, \text{AP}_{i-1}\}$ and how many $(\delta - \Delta)$ -bursts $(\mathcal{B}_{\Delta+1}, p_{\Delta+1}, \dots, \mathcal{B}_\delta, p_\delta)$ can occur over $\{\text{AP}_{i+1}, \dots, \text{AP}_n\}$. The value Δ describes the set $I(i) := \{0, \dots, k_i\} \cap \{\delta - k'_i, \dots, \delta\}$. As before, we can prove:

$$N_i^\Delta = \left(\sum_{i_\Delta=1}^{b_1} \sum_{i_{\Delta-1}=1}^{i_\Delta} \dots \sum_{i_2=1}^{i_3} \sum_{i_1=1}^{i_2} 1 \right) + \left(\sum_{i_{\delta-\Delta}=1}^{b_2} \sum_{i_{\delta-\Delta-1}=1}^{i_{\delta-\Delta}} \dots \sum_{i_2=1}^{i_3} \sum_{i_1=1}^{i_2} 1 \right)$$

where $b_1 = i - (\mathcal{B}_1 + \dots + \mathcal{B}_\Delta + (\Delta - 1))$ and $b_2 = \mathcal{B}_1 + \dots + \mathcal{B}_\Delta + \Delta + \xi - i + 1$. So, we can approximate $P_{\mathcal{M}}(P_i \text{ is authenticated})$ to:

$$\frac{\sum_{\Delta \in I(i)} \left(\sum_{i_\Delta=1}^{b_1} \sum_{i_{\Delta-1}=1}^{i_\Delta} \dots \sum_{i_2=1}^{i_3} \sum_{i_1=1}^{i_2} 1 + \sum_{i_{\delta-\Delta}=1}^{b_2} \sum_{i_{\delta-\Delta-1}=1}^{i_{\delta-\Delta}} \dots \sum_{i_2=1}^{i_3} \sum_{i_1=1}^{i_2} 1 \right)}{\sum_{i_\delta=1}^{\xi+1} \sum_{i_{\delta-1}=1}^{i_\delta} \dots \sum_{i_2=1}^{i_3} \sum_{i_1=1}^{i_2} 1} \quad (4)$$

We previously mentioned that Miner and Staddon [17] used p -random graphs to resist multiple bursts. Namely, they considered that the bursts occurring in the network can only have a finite number ℓ of pairwise different length $\mathcal{B}_1, \dots, \mathcal{B}_\ell$. They assumed that each burst of length \mathcal{B}_i can occur up to λ_i times. Their scheme was able to deal with any distribution of these $\delta := \lambda_1 + \dots + \lambda_\ell$ bursts. Here, we consider that each burst of length \mathcal{B}_i exactly occurs λ_i times. We denote L_δ the set of δ -length we can generate with this duplicating process. The cardinality of L_δ is the multinomial coefficient:

$$\binom{\delta}{\mathcal{B}_1 \dots \mathcal{B}_\ell}$$

Once again we assume that any δ -length has the same probability to be chosen by the opponent \mathcal{O} . We denote $\mathbb{P}(L_\delta)$ the set of permutations of L_δ . $\mathcal{B} := (\mathcal{B}_1, \dots, \mathcal{B}_1, \dots, \mathcal{B}_\ell, \dots, \mathcal{B}_\ell)$ is an element of L_δ (each \mathcal{B}_i is iterated λ_i times). We deduce the packet authentication probability provided by Lysyanskaya *et al.*'s scheme.

$$P_{\mathcal{M}}(P_i \text{ is authenticated}) = \left(\binom{\delta}{\mathcal{B}_1 \dots \mathcal{B}_\ell} \right)^{-1} \sum_{\tau \in \mathbb{P}(L_\delta)} P_{\tau(\mathcal{B})}(P_i \text{ is authenticated}) \quad (5)$$

where $P_{\mathcal{M}}(P_i \text{ is authenticated})$ is approximated by Formula (4) when \mathcal{M} is the loss pattern corresponding to the δ -length $\tau(\mathcal{B})$.

The efficiency of an authentication scheme can be defined as the smallest value of the packet authentication probability it provides. In other words, we are interested in $\min_{i \in \{1, \dots, n\}} P_{\mathcal{M}}(P_i \text{ is authenticated})$. Formula (4) and Equality (5) do not provide a clear lower bound on this minimal probability and therefore practical efficiency of the scheme is hard to guess. This motivates a search for a new authentication scheme exhibiting a clear authentication probability. This can be achieved by using LT codes as we will describe in the next session of this paper.

3 LT Codes for Multicast Stream Authentication

In this section, we will give a multicast authentication protocol using LT codes which is robust against packet loss and data injection. As in Section 2, we allow \mathcal{O} to use any pattern \mathcal{M} (not only the multiple-burst one) respecting the accuracy of (α, β) . Our technique also allows any new user to join the communication group at any block boundary and exhibits a lower bound for the packet authentication probability. We will first review the construction of LT codes. Then, we will develop our authentication scheme, prove its security and exhibit a minimal bound for the packet authentication probability.

3.1 Construction of 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 [12].

Encoding. We have a fixed number of input symbols denoted by I_1, \dots, I_n . In order to generate a new encoding symbol E , we use a probabilistic distribution called the Robust Soliton distribution to choose the degree⁴ d of the symbol E . We randomly pick d elements amongst the input symbols: I_{i_1}, \dots, I_{i_d} ⁵. 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 m encoding symbols E_1, \dots, E_m he first builds the bipartite graph used to compute E_1, \dots, E_m ⁶. 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 m is small and the encoding symbols have small degrees. At the beginning of the decoding process no I_j 's have been covered⁷. They are initialized with 0's. We first release⁸ all E_k '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^k 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^k . That is, the corresponding edges are removed from the graph.
3. For each N_j^k having one remaining neighbor in the new graph, N_j^k 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).

Step 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_n have been released from R . We will use the following theorem to deal with packet loss occurring during data transfer.

Theorem 2 ([12]) *For $\delta \in (0, 1)$, the decoding process fails with probability at most δ from any set of $m := n + (R + \frac{R}{2} + \dots + \frac{1}{n-R}) \ln(\frac{R}{\delta})$ encoding symbols where $R := c \ln(\frac{n}{\delta}) \sqrt{n}$ for a positive constant c determined within the Robust Soliton distribution.*

3.2 Our Authentication Protocol

We will consider the same opponent model as in Section 2 and the same definitions for (α, β) , h and the signature scheme (KeyGen, Sign, Verify). As said before, data is processed per block of n packets: P_1, \dots, P_n . We assume that the sender published a list of irreducible polynomials over \mathbb{F}_2 (remember that for any positive integer r , we can always build a irreducible polynomial of degree r over \mathbb{F}_2 [11]). On this public list, he also puts $\delta, n, \alpha, \beta, PK$ as well as h and the verification algorithm Verify. We denote τ_{par} the tag representing the communication parameters, namely: $\tau_{\text{par}} = n \|\alpha\| \|\beta\| \delta$. We assume that this tag is represented with a fixed number of bits b_{par} . We denote \mathcal{H} the size of a hash produced by h and \mathcal{S} the size of a signature. We first introduce the algorithm used by the sender. Here is a remark concerning Step 3. Since any element of \mathbb{F}_2^r can be represented as $\lambda_0 Y^0 + \lambda_1 Y^1 + \dots + \lambda_{r-1} Y^{r-1}$ where each λ_i belongs to \mathbb{F}_2 , we define the first 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 $(n-1)$.

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 Lysyanskaya *et al.*'s technique since the size of their field as well as the rate of their code have to be updated in case of modification 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.

We now justify our choices for \mathcal{N}, k, r and $P(X)$. In order to recover P_1, \dots, P_n with probability at least $1 - \delta$ despite loss, the integer \mathcal{N} must verify $\alpha \mathcal{N} \geq m$. In addition, we want to represent the hashes of these \mathcal{N} encoding symbols as coefficients of a polynomial $P(X)$ of degree k over an extension of degree r of \mathbb{F}_2 . We want k to be as large as possible

⁴Any LT code can be represented as a bipartite graph with I_1, \dots, I_n 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 conversely). We use the term *degree* to denote the number of neighbors a symbol has.

⁵This is how we build the bipartite graph representing the LT code.

⁶The positions of the input symbols XOR-ed to build an encoding symbol E_i are sent along with E_i [9].

⁷An input symbol I_j is said to be *covered* when it is the only adjacent vertex of an encoding symbol E_k . The covering operation is a XOR of the current value of I_j with E_k .

⁸A symbol is said to be *released* when we remove its representing vertex from the graph.

Algorithm 1 Authenticator

Input: The private key SK , the network rates α, β , a block $\{P_1, \dots, P_n\}$, its BID and the parameter δ .

1. Compute $\mathcal{N} = \begin{cases} \max(\lceil \frac{m}{\alpha} \rceil, \lceil \frac{\beta}{\alpha^2} \rceil) & \text{if } \frac{\beta}{\alpha^2} \notin \mathbb{N} \\ \max(\lceil \frac{m}{\alpha} \rceil, \frac{\beta}{\alpha^2} + 1) & \text{otherwise} \end{cases}$ where m is defined in Theorem 2. 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 along 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: $\sigma = \text{Sign}_{\text{SK}}(h(\text{BID} \| \tau_{\text{par}} \| h_1 \| \dots \| h_{\mathcal{N}}))$ and form the authentication tag $\tau = \tau_{\text{par}} \| h_1 \| \dots \| h_{\mathcal{N}} \| \sigma$. Compute $k = \begin{cases} \lfloor \frac{\alpha^2 \mathcal{N}}{\beta} \rfloor & \text{if } \frac{\alpha^2 \mathcal{N}}{\beta} \notin \mathbb{N} \\ \frac{\alpha^2 \mathcal{N}}{\beta} - 1 & \text{otherwise} \end{cases}$. Denote $\ell = k - \lceil \mathcal{N} \mathcal{H} + \mathcal{S} + b_{\text{par}} \rceil + 1$ and pad τ with ℓ zeros: $\tau' = \tau \| 0^\ell$.
3. Write τ' as the concatenation of $(k + 1)$ elements of \mathbb{F}_{2^r} : $p_0 \| \dots \| p_k$. Compute $r = \max\left(\lceil \log_2(\mathcal{N}) \rceil, \left\lceil \frac{\mathcal{H} \mathcal{N} + \mathcal{S} + b_{\text{par}}}{k+1} \right\rceil\right)$. Form the polynomial $P(X) = \sum_{i=0}^k p_i X^i$ and evaluate it in the first \mathcal{N} points of \mathbb{F}_{2^r} : $\forall i \in \{1, \dots, \mathcal{N}\} y_i = P(i)$.
4. Form the authenticated packets: $\forall i \in \{1, \dots, \mathcal{N}\} \text{AP}_i = \text{BID} \| i \| E_i \| N_i^1 \| \dots \| N_i^{d_i} \| y_i$

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

to minimize the extension degree r . The polynomial $P(X)$ will be evaluated in \mathcal{N} different positions. The receiver must solve the polynomial reconstruction problem to recover $P(X)$. In order to run GS-Decoder, the sufficient condition exhibited by Guruswami and Sudan [8] is to have $T > \sqrt{kN}$ where N is the number of points used for the reconstruction, T the number of these points (x, y) such that $y = P(x)$. Due to the definition of (α, β) , we have $T \geq \alpha \mathcal{N}$ and $\alpha \mathcal{N} \leq N \leq \beta \mathcal{N}$. Thus, $(T > \sqrt{kN})$ is verified as soon as $0 \leq k < \frac{\alpha^2 \mathcal{N}}{\beta}$. Since we want to split τ' into several elements, we have $k \geq 1$. Thus, $\frac{\alpha^2 \mathcal{N}}{\beta} > 1$ which justifies the value of \mathcal{N} at Step 1. The optimality of k at Step 2 follows.

Since $P(X)$ will be evaluated into \mathcal{N} points, we must have: $2^r \geq \mathcal{N}$. We want to represent τ as the concatenation $p_0 \| \dots \| p_k$ the size of which is $(k+1)r$ since each p_i is an element of \mathbb{F}_{2^r} . On the other hand, the size of τ is $\mathcal{N} \mathcal{H} + \mathcal{S} + b_{\text{par}}$. Thus, r must verify $(k+1)r \geq \mathcal{N} \mathcal{H} + \mathcal{S} + b_{\text{par}}$. Therefore, $r \geq \frac{\mathcal{N} \mathcal{H} + \mathcal{S} + b_{\text{par}}}{k+1}$. This justifies both choices of ℓ and r at Steps 2 and 3.

Now, we describe a variant of the GS-Decoder called *Modified GS-Decoder (MGS-Decoder)* which will be used as a subroutine of our decoding algorithm.

Algorithm 2 MGS-Decoder

Input: The number of packets per block n , the network rates α, β , the degree of the polynomial k and N elements $\{(x_i, y_i), 1 \leq i \leq N\}$.

1. If $N > \beta n$ or we have less than αn distinct values (x_i, y_i) then the algorithm stops.
2. Run GS-Decoder on the N points to get the list L of all polynomials of degree at most k over \mathbb{F}_{2^r} passing through at least αn of the N points.
3. Write the list L as: $L = \{L_1(X), \dots, L_{\mu}(X)\}$. Write each element of L as: $L_i(X) = \sum_{j=0}^k \mathcal{L}_{ij} X^j$ where $\forall i \in \{0, \dots, k\} \mathcal{L}_{ij} \in \mathbb{F}_q$. Form the elements: $\mathcal{L}_i = \mathcal{L}_{i0} \| \dots \| \mathcal{L}_{ik}$.

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

When the receiver gets data he first runs MGS-Decoder to build a list of elements (which are polynomial coefficients) and tests each of them until the signature is verified or the list exhausted. When the signature is recovered, the receiver knows the hashes of the original encoding symbols of the LT code. Then, he browses amongst the received packets to find as many original encoding symbols as he can. Due to the definition of α , there are at least $\alpha \mathcal{N}$ of them. Using the first $\alpha \mathcal{N}$ ones, he attempts to decode the LT code to recover all the original packets P_1, \dots, P_n . According to Theorem 2, this succeeds with probability at least $1 - \delta$. The formal description of this process is depicted as Algorithm 3.

Algorithm 3 Decoder

Input: The public key PK, the number of packets per block n , the network rates α, β , the block number BID, the parameter δ , the sender's list of irreducible polynomials and the set of received packets RP.

1. Compute \mathcal{N} . Write the packets as $\text{BID}_i \| j_i \| E_{j_i} \| N_{j_i}^1 \| \dots \| N_{j_i}^{d_{j_i}} \| y_{j_i}$ 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 $(N < \alpha n \text{ or } N > \beta n)$ then the algorithm stops.
2. Rename the set of received packets $\{\text{AP}'_1, \dots, \text{AP}'_N\}$ and write each element as: $\text{AP}'_i = \text{BID} \| j_i \| E_{j_i} \| N_{j_i}^1 \| \dots \| N_{j_i}^{d_{j_i}} \| y_{j_i}$ where $j_i \in \{1, \dots, \mathcal{N}\}$. Compute k and r . Get the irreducible polynomial of degree r from the sender's public list and run MGS-Decoder on the set $\{(j_i, y_{j_i}), 1 \leq i \leq N\}$ to get a list $\{c_1, \dots, c_\mu\}$ of candidates for signature verification. If MGS-Decoder rejects that set then the algorithm stops.

Compute ℓ . Initialize $h_i = \emptyset$ for $i \in \{1, \dots, \mathcal{N}\}$. Initialize $i = 1$. While the list has not been exhausted (and the signature not verified yet), we pick c_i . We first check if the ℓ last bits are zeros (we verify the length of the pad). If so, we write c_i as: $\tau_{\text{par}}^i \| h_1^i \| \dots \| h_{\mathcal{N}}^i \| \sigma^i$. If $\tau_{\text{par}}^i = \tau_{\text{par}}$ then check whether $\text{Verify}_{\text{PK}}(h(\text{BID} \| \tau_{\text{par}}^i \| h_1^i \| \dots \| h_{\mathcal{N}}^i), \sigma^i) = \text{true}$. In this case, we set $h_j = h_j^i$ for $j \in \{1, \dots, \mathcal{N}\}$ and break out the loop. In any other cases, we 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 AP'_i written as at Step 2, if $h(E_{j_i} \| N_{j_i}^1 \| \dots \| N_{j_i}^{d_{j_i}}) = h_\lambda$ then $E'_\lambda = E_{j_i}$, $d_\lambda = d_{j_i}$ and $\forall \xi \in \{1, \dots, d_{j_i}\} N'_\lambda^\xi = N_{j_i}^\xi$.

5. Pick the first $\alpha \mathcal{N}$ 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 E'_μ, \dots, E'^{μ} . Get n input symbols $\{P'_1, \dots, P'_n\}$ (where some of them can be empty).

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

3.3 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} . Similar to [14], we give the following definition:

Definition 5 (*KeyGen, Authenticator, Decoder*) is 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 to 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 $(\text{BID}, n, \alpha, \beta, \delta, \text{RP})$.

\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 $\alpha \mathcal{N}$ packets of some authenticated packet set AP_i for block identification tag BID, decoding failure probability δ and parameters n, α, β , 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 .

The difference from the definition given in [14] 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 $\alpha \mathcal{N}$ original elements, the whole original set $\{P_1, \dots, P_n\}$ is recovered with some probability. Nevertheless, Definition 5 involves that no incorrect packets can be authenticated. That is: $\forall i \in \{1, \dots, n\} P'_i \in \{\emptyset, P_i\}$ where P'_i denotes the i^{th} packet output by Decoder. Lysyanskaya *et al.* showed that their scheme is secure and (α, β) -correct. Following their arguments, we obtain the following result for ours.

Theorem 3 *Our scheme* (*KeyGen, Authenticator, Decoder*) *is secure and* (α, β) -*correct.*

Proof.

[Sketch] If the scheme is neither secure nor (α, β) -correct then \mathcal{O} is able to create data packets which will be authenticated by the receiver after LT decoding at step 5. Nevertheless the LT decoding process is consistent. That is, if only correct encoding symbols are given to the decoder then it only outputs the corresponding input symbols (along with some empty symbols when the decoding process is not successful). Therefore, \mathcal{O} is able to create (at least) one fake symbol $\tilde{E}_i \| \tilde{N}_i^1 \| \dots \| \tilde{N}_i^{d_i}$ such that its hash \tilde{h}_i is a part of the element \tilde{c} which successfully verified the signature at Step 3. Since

h is collision resistant, we have: $\forall j \in \{1, \dots, \mathcal{N}\} \tilde{h}_i \neq h_j$. Thus, \tilde{c} was never signed by the sender and \mathcal{O} is able to break the signature scheme. Due to space limitations, we did not include the complete proof here. Note that, it exhibits the necessity of using τ_{par} as a part of the authentication tag. \square

Thus, our authentication scheme is as secure and correct as the technique developed in [14]. We will now study the packet authentication probability of our protocol.

3.4 Analysis of the Packet Authentication Probability

We now justify our use of LT codes to enable the receivers to recover all the n data packets P_1, \dots, P_n despite loss with probability close to 1 as claimed in Section 1. We assume that the receiver gets a set RP of packets. Since (α, β) is the network accuracy we have $|\text{RP}| \leq \beta\mathcal{N}$ and at least $\alpha\mathcal{N}$ original authenticated packets are amongst RP. As before, we denote \mathcal{M} the modification pattern used by \mathcal{O} . We fix i in $\{1, \dots, n\}$. To be more concise, we denote V_{RP} the probabilistic event $\{\text{RP verifies the signature}\}$. Using Bayes' theorem, we get:

$$\begin{aligned} p_{\mathcal{M}}(P_i \text{ is authenticated} | V_{\text{RP}}) p_{\mathcal{M}}(V_{\text{RP}}) \\ = \\ p_{\mathcal{M}}(V_{\text{RP}} | P_i \text{ is authenticated}) p_{\mathcal{M}}(P_i \text{ is authenticated}) \end{aligned}$$

Again, even if one of the events $\{P_i \text{ is authenticated}\}$ or V_{RP} is $p_{\mathcal{M}}$ -negligible the previous equality is still true since both products would be 0. Due to the design of Decoder, a necessary condition to output packets is to verify the signature. So: $p_{\mathcal{M}}(V_{\text{RP}} | P_i \text{ is authenticated}) = 1$. On the other hand, since (α, β) is accurate RP always verifies the signature since MGS-Decoder outputs the list of all polynomials passing through at least $\alpha\mathcal{N}$ of the elements of RP. Thus, the polynomial used by the sender belongs to that list and therefore the signature is verified. So: $p_{\mathcal{M}}(V_{\text{RP}}) = 1$. Thus, we get: $p_{\mathcal{M}}(P_i \text{ is authenticated}) = p_{\mathcal{M}}(P_i \text{ is authenticated} | V_{\text{RP}})$ which can be written as:

$$\begin{aligned} p_{\mathcal{M}}(P_i \text{ is authenticated}) &= p_{\mathcal{M}}(\text{All packets are authenticated} | V_{\text{RP}}) \\ &+ \\ p_{\mathcal{M}}(\{P_i \text{ is authenticated}\} \cap \{\text{At least one } P_j \text{ is not authenticated}\} | V_{\text{RP}}) \end{aligned}$$

Since $p_{\mathcal{M}}(V_{\text{RP}}) = 1$, we deduce:

$$p_{\mathcal{M}}(\text{All packets are authenticated} | V_{\text{RP}}) = p_{\mathcal{M}}(\text{The LT code is successfully decoded})$$

In addition, we have:

$$\begin{aligned} p_{\mathcal{M}}(\{P_i \text{ is authenticated}\} \cap \{\text{At least one } P_j \text{ is not authenticated}\} | V_{\text{RP}}) \\ = \\ p_{\mathcal{M}}(\{P_i \text{ is authenticated}\} \cap \{\text{The LT code is not successfully decoded}\}) \end{aligned}$$

The last event is not $p_{\mathcal{M}}$ -negligible in general since any symbol released from the ripple during the LT decoding process is consistent with the original input symbols [12]. Thus:

$$p_{\mathcal{M}}(P_i \text{ is authenticated}) \geq p_{\mathcal{M}}(\text{The LT code is successfully decoded}) \geq 1 - \delta$$

Since this is true for any value i , we deduce that:

$$\min_{i \in \{1, \dots, n\}} p_{\mathcal{M}}(P_i \text{ is authenticated}) \geq 1 - \delta$$

We also notice that this lower bound does not depends on the modification pattern \mathcal{M} .

4 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 computation of the lower bound on the packet authentication probability 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.* [9]. In their work, they changed the construction of LT codes given by Luby [12] 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 [12], 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 [12]. This is of particular interest in our case (Step 4 of Authenticator) 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 [9, 12, 15, 26], we built Table 1. Both Raptor and Online codes require preprocessing of data before encoding. In [15], Maymounkov proposed two different ways to do so for Online codes. The complexities shown in Table 1 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	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 1: Complexity comparison for different classes of rateless codes.

According to Table 1, 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 [13, 26]. Given our opponent model, it is unlikely that BEC can be the modification pattern used by \mathcal{O} . Nevertheless, a recent work by Palanki and Yedidia [18] 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.* [5] 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.

As suggested by Harrelson *et al.* [9], it is possible to reduce the size of information to be transmitted and achieve a constant packet overhead at the cost of extra symbols for decoding (see Table 1). Since achieving a regular throughput within the communication channel avoids data congestion, substituting original LT codes by their modifications in our authentication protocol is recommended (the value of m in Theorem 2 has to be updated consequently). 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 3 will be even more efficient when substituting LT codes by Raptor codes (exhibiting the same modifications for their internal LT coding).

Nevertheless, these threshold values enabling recovery of the n data packets can still be too important for some applications. Karp *et al.* [10] 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. The sender computes the number of packets he has to transmit in order to achieve at most this rate of non-recovered packets. In this case, the lower bound computed on Section 3 is not valid any longer but the security and correctness of the scheme are still preserved.

5 Conclusion

In 2003, Lysyanskaya *et al.* [14] designed a multicast authentication scheme dealing with both packet loss and data injection. Unfortunately, its packet authentication probability does not exhibit an easy lower bound and when a packet is dropped during transmission its content is definitely lost for the receivers. Our technique, which can be considered as an extension of theirs, has two main advantages. First, it allows the receivers to reconstruct erased data which, to our knowledge, had never been achieved yet by any existing multicast stream authentication protocol using signature

dispersion. Second, it exhibits a minimal lower bound on the packet authentication probability which can be chosen arbitrary close to 1. Our reconstruction property relies on the fact that (α, β) is the network accuracy which can be hard to determine when the communication group size is large. Hopefully any couple $(\tilde{\alpha}, \tilde{\beta})$ such that $\tilde{\alpha} \leq \alpha$ and $\beta \leq \tilde{\beta}$ will also be fine for our scheme. This is at the cost of creating more encoding symbols to run GS-Decoder. Thus, this couple of parameters has to be chosen carefully in order to respect the heterogeneity of the receivers without generating unnecessary data. Our scheme also allows new users to join the communication group at any time since each block of n packets contains its own signature. We also proposed to use a modified version of LT codes to achieve reasonable and fixed overhead per packet preventing the network from having too irregular variations of data flow. Given [5, 18], we also stress that Raptor codes could provide good implementations of our scheme if used instead of LT codes.

Acknowledgment

The authors would like to thank Professor Josef Pieprzyk for valuable conversations as well as 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 Project DP0344444. The first author's work was also funded by an iMURS scholarship supported by Macquarie University.

References

- [1] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology - Crypto'92*, volume 740 of *Lecture Notes in Computer Science*, pages 471 – 486, Santa Barbara, USA, August 1992. Springer - Verlag.
- [2] 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.
- [3] 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.
- [4] Yvo Desmedt, Yair Frankel, and Moti Yung. Multi-receiver/multi-sender network security: Efficient authenticated multicast/feedback. In *INFOCOM '92*, volume 3, pages 2045 – 2054, Florence, Italy, May 1992. IEEE Press.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and their Applications - Revised Edition*. Cambridge University Press, 2000.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] Petar Maymounkov. Online codes. Technical report, New York University, November 2002.

- [16] 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.
- [17] 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.
- [18] Ravi Palanki and Jonathan S. Yedidia. Rateless codes on noisy channels. In *38th Annual Conference on Information Sciences and Systems*, Princeton, USA, March 2004.
- [19] 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.
- [20] Vern Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277 – 292, June 1999.
- [21] Adrian Perrig and J. D. Tygar. *Secure Broadcast Communication in Wired and Wireless Networks*. Kluwer Academic Publishers, 2003.
- [22] Josef Pieprzyk, Thomas Hardjono, and Jennifer Seberry. *Fundamentals of Computer Security*. Springer, 2003.
- [23] Malempati Madhusudana Rao. *Conditional Measures and Applications (Second Edition)*. CRC Press, 2005.
- [24] 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.
- [25] Rei Safavi-Naini and Huaxiong Wang. New results on multi-receiver authentication code. In *Advances in Cryptology - Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 527 – 541, Espoo, Finland, June 1998. Springer - Verlag.
- [26] Amin Shokrollahi. Raptor codes. Technical report, Digital Fountain, June 2003.
- [27] Douglas R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 1995.
- [28] 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.
- [29] Chung Kei Wong and Simon S. Lam. Digital signatures for flows and multicasts. *IEEE/ACM Transactions on Networking*, 7(4):502 – 513, August 1999.
- [30] 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.