

Achieving Multicast Stream Authentication using MDS Codes*

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 address the multicast stream authentication problem when the communication channel is under the control of an opponent who can drop, reorder or inject data. In such a network model, packet overhead and computing efficiency are important parameters to be taken into account when designing a multicast authentication protocol. Our construction will exhibit three main advantages. First, our packet overhead will only be a few hashes long. Second, we will exhibit a number of signature verifications to be performed by the receivers which will turn to be $O(1)$. Third, every receiver will still be able to recover all the data packets emitted by the sender despite losses and injections occurred during the transmission of information.

Keywords: Stream Authentication, Polynomial Reconstruction, Erasure Codes.

1 Introduction

Broadcast communication is an essential mechanism to disseminate digital media from a single sender to a large audience via a public channel such as the Internet. The applications cover a broad area including digital radio, air traffic control as well as software updates for instance. Unfortunately, large-scale multicasts prevent lost content from being redistributed since the loss of any piece of the data stream can cause a flood of retransmission requests at the sender. In addition, the network can be under the control of malicious users performing harmful actions on the data stream. Therefore, the security of a broadcast protocol relies on two aspects: the network properties and the opponents' computational power. Investigations concerning unconditionally secure protocols have been made in [5, 7, 37]. Unfortunately, either these schemes can only be used for a single authentication or they have too large storage requirements for practical applications. In this work, we will consider that the opponents have bounded computational powers.

Applications like digital TV or stock quotes suggest that the data stream can be large and eventually infinite. Nevertheless, receivers must be able to authenticate collected information within a short period of delay upon reception. Since many protocols will transfer private or sensitive content, non-repudiation of the sender is required for most of them. Unfortunately, signing each packet¹ is impracticable as digital signatures are generally time expensive. In addition, bandwidth limitations prevent k -time signatures [35] from being used due to their large size. That is why a general 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.* [31, 32], Golle and Modadugu [9] and Miner and Staddon [23] 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 [8, 30, 42] and provided bounds on the packet authentication probability. Gao and Yao [44] 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. Since networks like the Internet only provide a best effort delivery, it narrows the range of applications of these schemes.

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 [1, 26, 27, 28, 29] but none of them tolerates a single packet injection. In 2003, Lysyanskaya *et al.* [19] designed a technique (called in this paper LTT) resistant to packet

*The original version of this paper appears in the proceedings of the 5th International Conference on Cryptology and Network Security (CANS 2006), Lecture Notes in Computer Science, vol. 4301, pp 108 - 125, Springer - Verlag.

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

loss and data injections using Reed-Solomon codes [34] 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 . Unfortunately, that approach does not allow lost content to be recovered. This drawback was also present in Tartary and Wang’s scheme [40]. This problem was overcome by Karlof *et al.* in 2004. In [15], they designed a protocol called PRABS using an erasure code (to recover data loss) along with a one-way accumulator [3, 4, 24, 25] based on a Merkle hash tree [22] (to deal with injections). Their approach is similar to Wong and Lam’s construction but it has the advantages of allowing recovery of all original data packets with only $O(1)$ signature verifications to compute. Unfortunately, PRABS’s augmented packets³ must still carry $\lceil \log_2 n \rceil$ hashes.

In this paper, we propose a scheme having the advantages of the previous two constructions without their drawbacks. Every single original data packet will be recovered by any receiver and, contrary to [19] where only an asymptotic study was performed, we will exhibit an upper bound on the number of signature verifications to be executed per block. This bound will be valid for any block length and will turn to be $O(1)$ as in [15, 19]. 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. To the best of our knowledge, PRABS is the only multicast stream authentication protocol ever designed which exhibits the recovery property and such a complexity value for signature verifications at the same time. As said earlier, this is at the cost of a logarithmic number of hashes appended to each packet. Our recovery capacity will be due to a Maximum Distance Separable (MDS) code construction developed by Lacan and Fimes in 2004. Based on their work [16], we will enlighten 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 in [19]) 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 in [15] for practical applications. This is a major concern in multicast communication since too large packets may involve irregular throughput of data and sometimes result in congestion of the network information flow.

This paper is organized as follows. In Section 2, we will present our network model as well as a few results from [15, 19]. In Section 3, we will describe our authentication scheme. Its security and recovery property will be studied in Section 4. In Section 5, we will compare our scheme to those from [15, 19] as our work can be seen as their extension. Finally, we will summarize our contribution 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 code construction by Lacan and Fimes. Finally, we will recall an algorithm by Guruswami and Sudan achieving polynomial reconstruction. As in [19, 40], it will be used to deal with packet injections.

2.1 Network Model

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 scheme, 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 at least $\lceil \alpha n \rceil$ original augmented packets are received amongst a total which does not exceed $\lfloor \beta n \rfloor$ elements.

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 [41]. In the second case there are no packet injections from \mathcal{O} . Thus, using an erasure code (as in [6] for instance) is enough to recover P_1, \dots, P_n . Therefore, in this work, we will only study the case: $0 < \alpha < 1 < \beta$.

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.

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

2.2 Code Construction

In this paper, we will 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: $D - 1 \leq N - K$ [20]. It is known that any $[N, K, D]$ code can correct up to $D - 1$ erasures [43]. 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 [20].

Now, we describe the MDS code construction developed in [16]. 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 [17]. Operations in \mathbb{F}_{2^q} are performed modulo a polynomial $Q(X)$ of degree q which is irreducible over \mathbb{F}_2 . From [16], we have:

Theorem 1 *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.

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 represents r elements of the field \mathbb{F}_{2^q} . The choice of the efficiency parameter r will be explained in Section 3.

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: $S_i := S_i^1 \| \dots \| S_i^r$. 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: $c_j := (c_1^j \dots c_N^j)$. Build the N modified symbols as: $\forall j \in \{1, \dots, N\} \hat{S}_j := c_1^j \| \dots \| c_r^j$.

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: $\forall i \in \{1, \dots, K\} \hat{S}_{j_i} = c_1^{j_i} \| \dots \| c_r^{j_i}$ and write: $\forall i \in \{1, \dots, r\} c'_i := (c_1^{j_i} \dots c_K^{j_i})$.
2. Build G' as the restriction of G to columns j_1, \dots, j_K . Compute r messages as: $\forall i \in \{1, \dots, r\} m_i := c'_i G'^{-1}$.
3. Write each message as: $m_j := (m_1^j \dots m_K^j)$ where each m_i^j is $\deg(Q(X))$ bits long. Recover the K symbols as: $\forall i \in \{1, \dots, K\} S_i = m_1^i \| \dots \| m_r^i$.

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

Notice that the polynomial $Q(X)$ is used at Step 2 of Algorithm 1 and Algorithm 2 when performing matrix multiplications.

2.3 Polynomial Reconstruction Algorithm

In [13], 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 $\tilde{Q}(X)$ the polynomial used to perform operations in that field.

Algorithm 3 MPR

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

1. If there are no more than \sqrt{KN} distinct points then the algorithm stops.
2. Using $\tilde{Q}(X)$, run Poly-Reconstruct on the T points to get the list of all polynomials of degree at most K over $\mathbb{F}_{2^{\tilde{q}}}$ (where $\tilde{q} = \deg(\tilde{Q}(X))$) passing through at least N of the previous points.
3. Write the list $\{L_1(X), \dots, L_\mu(X)\}$ and each element: $L_i(X) := \mathcal{L}_{i0} + \dots + \mathcal{L}_{iK}X^K$ where $\forall i \in \{0, \dots, K\} \mathcal{L}_{ij} \in \mathbb{F}_{2^{\tilde{q}}}$. Form the elements: $\mathcal{L}_i := \mathcal{L}_{i0} \parallel \dots \parallel \mathcal{L}_{iK}$.

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

3 Our Construction

We first give a global overview of our authentication scheme. As in [15, 19], we need a collision-resistant hash function h [33] and a secure signature scheme $(\text{Sign}_{\text{SK}}, \text{Verify}_{\text{PK}})$ [39] the key pair of which (SK, PK) is created by a generator KeyGen .

3.1 Informal Scheme Description

From the n data packets P_1, \dots, P_n we want to generate n augmented packets $\text{AP}_1, \dots, \text{AP}_n$ 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 in [19], we build a polynomial $A(X)$ of degree at most ρn (for some rational constant ρ), the coefficients of which represent $h_1 \parallel \dots \parallel h_n \parallel \sigma$ (σ is the signature). We build the augmented packets as: $\forall i \in \{1, \dots, n\} \text{AP}_i := \text{BID} \parallel i \parallel \hat{S}_i \parallel A(i)$ where BID represents the position of the block P_1, \dots, P_n within the whole data stream.

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 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 .

3.2 Formal Scheme Construction

For our construction, we assume that α and β are rational numbers (see Section 4). Thus, we can represent them over a finite number of bits using their numerator and denominator. Denote \mathcal{P} the bit size of the data packets. In order to have $\lceil \alpha n \rceil$ symbols of equal length we must pad $P_1 \parallel \dots \parallel P_n$ with ℓ zeros appropriately (see Step 1 of Authenticator). Then, we can split $P_1 \parallel \dots \parallel P_n \parallel 0^\ell$ into n symbols $S_1, \dots, S_{\lceil \alpha n \rceil}$ where each S_i is m bits long with $m := \lceil \frac{n\mathcal{P}}{\lceil \alpha n \rceil} \rceil$. The efficiency parameter r is chosen by the sender as a divisor of m such that the receivers can efficiently perform computations over the field \mathbb{F}_{2^q} where $q := \frac{m}{r}$. Notice that the smaller r is, the larger the number of messages to be encoded at Step 2 of EncodeMDS is.

In order to run Poly-Reconstruct as a part of MPR, we have to choose $\rho \in (0, \frac{\alpha^2}{\beta})$. Notice that ρ has to be rational since ρn is an integer. Table 1 summarizes the scheme parameters which are assumed to be publicly known.

n : Block length	$\mathcal{Q}(X)$: Polynomial representing the field for the MDS code
\mathcal{P} : Packet size (in bits)	G : Generating matrix of the MDS code
α, β : Network rates	r : Efficiency parameter of the MDS code
ρ : Ratio	$\tilde{Q}(X)$: Polynomial representing the field for polynomial interpolation

Table 1: Public parameters for our authentication 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 while the digital signature is RSA-1024. We denote \mathcal{H} the digest bit length and s the bit length of a signature. Since h and the digital signature are publicly known, so are \mathcal{H} and s .

The sender uses Algorithm 4 to construct the augmented packets. Here is a remark concerning Step 5. Since any element of $\mathbb{F}_{2^{\tilde{m}}}$ can be represented as $\lambda_0 Y^0 + \lambda_1 Y^1 + \dots + \lambda_{\tilde{m}-1} Y^{\tilde{m}-1}$ 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 $(n-1)$.

Algorithm 4 Authenticator

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

1. Compute: $b = n\mathcal{P} \bmod \lceil \alpha n \rceil$. Denote ℓ as $\ell = (0 \text{ if } b = 0) \text{ or } (\lceil \alpha n \rceil - b \text{ otherwise})$. Write $P_1 \| \dots \| P_n \| 0^\ell$ as $S_1 \| \dots \| S_{\lceil \alpha n \rceil}$ where each S_i is $m = \lceil \frac{n\mathcal{P}}{\lceil \alpha n \rceil} \rceil$ bits long.
2. Compute: $(\hat{S}_1, \dots, \hat{S}_n) = \text{EncodeMDS}(n, \lceil \alpha n \rceil, \mathcal{Q}(X), G, S_1, \dots, S_{\lceil \alpha n \rceil}, r)$.
3. Compute: $\forall i \in \{1, \dots, n\} h_i = h(\hat{S}_i)$ and form the string $h_1 \| \dots \| h_n \| \sigma$ where $\sigma = \text{Sign}_{\text{SK}}(h(\text{BID} \| \alpha \| \beta \| n \| \mathcal{P} \| h_1 \| \dots \| h_n))$.
4. Compute: $\tilde{b} = n\mathcal{H} + s \bmod (\rho n + 1)$. Denote $\tilde{\ell}$ as $\tilde{\ell} = (0 \text{ if } \tilde{b} = 0) \text{ or } (\rho n - 1 - \tilde{b} \text{ otherwise})$. Write $h_1 \| \dots \| h_n \| \sigma \| 0^{\tilde{\ell}}$ as $a_0 \| \dots \| a_{\rho n}$ where each a_i is $\tilde{m} = \lceil \frac{n\mathcal{H} + s}{\rho n + 1} \rceil$ bits long. Build the polynomial $A(X) = a_0 + a_1 X + \dots + a_{\rho n} X^{\rho n}$.
5. Using $\tilde{\mathcal{Q}}(X)$, evaluate $A(X)$ at the first n elements of $\mathbb{F}_{2^{\tilde{m}}}$. Using the n couples $(i, A(i))$, build the n augmented packets as: $\forall i \in \{1, \dots, n\} \text{AP}_i = \text{BID} \| i \| \hat{S}_i \| A(i)$.

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

We have two remarks which will be important when proving the security of our scheme in Section 4. First, when assuming that α and β were rational in Section 2, we claimed that we could represent them over a finite number of bits. This allows us to include the concatenation $\alpha \| \beta$ as a part of the string to be hashed in Step 3. Second it should be pointed out that when $(n, \alpha, \beta, \mathcal{P}, \rho)$ are given, every single step of Authenticator is uniquely determined as soon as $(\mathcal{Q}(X), G, r, \tilde{\mathcal{Q}}(X))$ are provided. Furthermore, since the rational ρ only depends on α, β and n , it is realistic to presume that when (n, α, β) are given, ρ is also uniquely determined. Thus, we can assume for our scheme that when $(n, \alpha, \beta, \mathcal{P})$ are given, $(\rho, \mathcal{Q}(X), G, r, \tilde{\mathcal{Q}}(X))$ are uniquely determined.

Notice that, to perform Step 5, we must have at least n distinct elements in $\mathbb{F}_{2^{\tilde{m}}}$. It can be shown that this property is verified for n up to 2^{20} . In practical applications, however, n will be roughly 1000 ($\simeq 2^{10}$) (see [32] for instance). We refer the reader interested in the details of this affirmation to Appendix A. The receiver identifies packets using Algorithm 5.

Algorithm 5 Decoder

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

1. Write the packets as $\text{BID}_i \| j_i \| S'_{j_i} \| A_{j_i}$ and discard those having $\text{BID}_i \neq \text{BID}$ or $j_i \notin \{1, \dots, n\}$. Denote \mathcal{N} the number of remaining elements. If $(\mathcal{N} < \lceil \alpha n \rceil \text{ or } \mathcal{N} > \lfloor \beta n \rfloor)$ then the algorithm stops.
2. Rename the remaining elements as $\{\text{AP}'_1, \dots, \text{AP}'_{\mathcal{N}}\}$ and write each element as: $\text{AP}'_i = \text{BID} \| j_i \| S'_{j_i} \| A_{j_i}$ where $j_i \in \{1, \dots, n\}$. Run MPR on the set $\{(j_i, A_{j_i}), 1 \leq i \leq \mathcal{N}\}$ to get a list $\{\mathcal{C}_1, \dots, \mathcal{C}_\mu\}$ of candidates for signature verification. If MPR rejects that set then the algorithm stops.
3. Set $h_k = \emptyset$ for $k \in \{1, \dots, n\}$. Compute $\tilde{\ell}$ as in Step 4 of Authenticator. Set $i = 1$. While $\{\mathcal{C}_1, \dots, \mathcal{C}_\mu\}$ has not been exhausted, pick \mathcal{C}_i . Check if \mathcal{C}_i ends with $\tilde{\ell}$ zeros. If so, write \mathcal{C}_i as: $h_1^i \| \dots \| h_n^i \| \sigma^i \| 0^{\tilde{\ell}}$ where each h_k^i is \mathcal{H} bits long. If $\text{Verify}_{\text{PK}}(h(\text{BID} \| \alpha \| \beta \| n \| \mathcal{P} \| h_1^i \| \dots \| h_n^i), \sigma^i) = \text{TRUE}$ then set $h_k = h_k^i$ for $k \in \{1, \dots, n\}$ and break out the loop. In any other cases, 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 AP'_i written as at Step 2, if $h(S'_{j_i}) = h_\lambda$ then $\hat{S}_\lambda = S'_{j_i}$.
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: $(S_1, \dots, S_{\lceil \alpha n \rceil}) = \text{DecodeMDS}(n, \lceil \alpha n \rceil, \mathcal{Q}(X), G, (p_1, \hat{S}_{p_1}), \dots, (p_\gamma, \hat{S}_{p_\gamma}), r)$.
6. Set $P'_k = \emptyset$ for $k \in \{1, \dots, n\}$. Compute ℓ as in Step 1 of Authenticator. If the last ℓ bits of $S_1 \| \dots \| S_{\lceil \alpha n \rceil}$ are not zeros then the algorithm stops. Otherwise, write that concatenation as $P'_1 \| \dots \| P'_n \| 0^\ell$ where each P'_i is \mathcal{P} bits long.

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

4 Security and Recovery Analysis

4.1 Security of the Scheme

Similarly to [19], we give the following definition:

Definition 1 ($\text{KeyGen}, \text{Authenticator}, \text{Decoder}$) is a **secure** and (α, β) -**correct** 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 $(\text{BID}, n, \alpha, \beta, \mathcal{P}, \rho, \mathcal{Q}(X), \tilde{\mathcal{Q}}(X), G, r, \text{RP})$

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

- (a) (correctness violation) \mathcal{O} succeeds to output RP such that even if it contains $\lceil \alpha n \rceil$ packets (amongst a total number of elements which does not exceed $\lfloor \beta n \rfloor$) for some block identification tag BID , Decoder fails at identifying all the correct packets.
- (b) (security violation) \mathcal{O} succeeds to output RP such that Decoder outputs $\{P'_1, \dots, P'_n\}$ that was never authenticated by Authenticator for parameters $(\text{BID}, n, \alpha, \beta, \mathcal{P}, \rho, \mathcal{Q}(X), \tilde{\mathcal{Q}}(X), G, r)$.

Lysyanskaya *et al.* proved that their construction satisfied the previous conditions of security and correctness. We have a similar result for our scheme.

Theorem 2 Our scheme $(\text{KeyGen}, \text{Authenticator}, \text{Decoder})$ is **secure** and (α, β) -**correct**.

Proof.

[Sketch] If the scheme is neither secure nor (α, β) -correct then \mathcal{O} is able to generate data packets which will be authenticated by the receiver after MDS decoding at Step 5 of Decoder . Nevertheless, the decoding algorithm of the MDS code is consistent. This means that if only correct elements $\hat{S}_{p_1}, \dots, \hat{S}_{p_\gamma}$ are given to DecodeMDS then it outputs the corresponding original elements $S_1, \dots, S_{\lceil \alpha n \rceil}$. This consistency involves that \mathcal{O} can generate (at least) one incorrect symbol S'_i for some $i \in \{1, \dots, n\}$ such that its hash h'_i is a part of the element C' which successfully verified the signature at step 3 of Decoder . Since h is collision resistant, we have: $\forall j \in \{1, \dots, n\} h'_i \neq h_j$. Thus, C' was never signed by the sender and \mathcal{O} is able to forge the signature scheme. Due to space limitations, we did not include the complete proof here. Note that, it exhibits the necessity of using $\alpha \|\beta\|n \|\mathcal{P}$ as a part of the element to be hashed at step 3 of Authenticator . \square

4.2 Recovery Property

We will now demonstrate that our scheme enables any receiver to recover the n data packets (as in [15]) and the number of signature verifications to be performed per block is $O(1)$ (as in [19]). We also provide a non-asymptotic bound on this number of verifications. First we introduce the following definition:

Definition 2 We say that the survival and flood rates (α, β) are **accurate** to the network for a flow of n symbols if:

1. Data are sent per block of n elements through the network.
2. For any block of n elements $\{E_1, \dots, E_n\}$ emitted by the sender, if we denote $\{\tilde{E}_1, \dots, \tilde{E}_\mu\}$ the set of received packets then $\mu \leq \lfloor \beta n \rfloor$ and at least $\lceil \alpha n \rceil$ elements of $\{E_1, \dots, E_n\}$ belong to $\{\tilde{E}_1, \dots, \tilde{E}_\mu\}$.

The second condition must be true for each receiver belonging to the communication group.

Notice that, when n is fixed, (α, β) is not unique since any $(\tilde{\alpha}, \tilde{\beta})$ with $\tilde{\beta} \geq \beta$ and $0 < \tilde{\alpha} \leq \alpha$ is also accurate for the same flow n . That is why we can always choose α, β as rational numbers as claimed in Section 3. From now on, we assume that (α, β) is accurate for our network flow n . It should be pointed out that the asymptotic result from [19] was obtained under the same assumption. Despite PRABS can tolerate an unbounded number of injections, it still requires a minimal number of original elements to be received in order to enable recovery of the n data packets P_1, \dots, P_n . Thus, assuming the accuracy of (α, β) to compare our scheme to [15] seems to be a realistic hypothesis as well. We now present our main theorem the proof of which can be found in Appendix B.

Theorem 3 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 $U(n) := \min(\lfloor U_1(n) \rfloor, \lfloor U_2(n) \rfloor)$ where:

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

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

It should be noticed that when n is fixed, the value $U(n)$ increases when ρ gets closer to the threshold value $\frac{\alpha^2}{\beta}$ or to 0. In addition, if ρ is too small then the size of the field \mathbb{F}_{2^m} gets larger which can result in a prohibitive computational cost for some of the receivers. Therefore, the sender must pay attention to the choice of ρ when setting up the scheme in order to have a balanced trade-off between the efficiency of field operations and a reasonable number of signature verifications to perform.

To illustrate this remark we computed $U(n)$ when $n = 1000$ as in [32]. $U(1000)$ was evaluated for values 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 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 . We also implemented $V(1000)$ where $V(n)$ is the bound provided by Karlof *et al.* [15] (see Section 5 for more details). The results are shown in Table 2.

	(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 3	55 3	66 4	88 5	21 2	24 2	29 3	39 3
30%	20 3	23 3	28 4	38 5	9 2	10 2	12 3	16 3
50%	17 3	20 3	24 4	31 5	7 2	8 2	10 3	13 3
70%	20 3	23 3	28 4	38 5	9 2	10 2	12 3	15 3
90%	48 3	55 3	66 4	88 5	21 2	24 2	28 3	36 3
	(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 2	21 2	25 2	34 3	15 2	17 2	20 2	27 3
30%	8 2	9 2	11 2	14 3	6 2	7 2	8 2	11 3
50%	6 2	7 2	9 2	11 3	5 2	6 2	7 2	9 3
70%	8 2	9 2	10 2	13 3	6 2	7 2	8 2	10 3
90%	19 2	21 2	25 2	31 3	15 2	16 2	19 2	24 3

Table 2: Evaluations of $U(1000)|V(1000)$.

5 Efficiency Comparison

Since our technique is an extension of [15, 19], we will enlighten the benefits our construction provides in this section.

5.1 Signature Verification Complexity

We saw in Theorem 3 that the asymptotic behavior of $U(n)$ is identical to the asymptotic result from [19]. According to Theorem 3 from [15], the number of signature verifications for PRABS is upper bounded by $V(n) := \left\lceil \frac{\beta n}{\alpha n} \right\rceil + 1$. Since $V(n) \leq \frac{\beta}{\alpha} + 1$, it is clear that $V(n)$ is $O(1)$ and $V(n) \leq \lfloor U_1(n) \rfloor$. Table 2 clearly shows that $V(n)$ is much smaller than $U(n)$. 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 said earlier, this is impractical since such large packets can cause congestion in the network throughput. Table 2 results suggest that $U(n)$ is minimal when ρ is roughly half the threshold value $\frac{\alpha^2}{\beta}$.

5.2 Packet Overhead

Our augmented packets are written as $\text{BID}||i||\hat{S}_i||A(i)$. The element $A(i)$ is $\left\lceil \frac{n\mathcal{H}+s}{n\rho+1} \right\rceil$ bits long (as in [19]) and \hat{S}_i is slightly larger than P_i since it is $\left\lceil \frac{nP}{\lceil \alpha n \rceil} \right\rceil$ bits long. Thus our augmented packet size is slightly larger than those from [19]. 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 $\mathbb{F}_{2^{\tilde{m}}}$ must contain at least n points for polynomial evaluation. Therefore, their construction requires: $n \leq 2^{\tilde{m}}$. Since $2^{\tilde{m}} \in O(1)$, it is clear that the previous inequality is not satisfied for large values of n . Thus, it is mathematically impossible to study the asymptotic behavior of packet overhead since LTT is not constructible for large values of n . For practical realizations (i.e. realistic values of n), the previous inequality is satisfied (see Appendix A) and then it makes sense to approximate the overhead bit size to $\frac{\mathcal{H}}{\rho}$.

For the same reason, our comparison to PRABS is restricted to practical values of n . In both constructions, the n data packets are encoded to recover from a fraction $1 - \alpha$ of erasures. Thus, PRABS augmented packets can be written as $\text{BID}||i||\tilde{S}_i||H(i)$ where $H(i)$ is the concatenation of $\lceil \log_2 n \rceil$ hashes and the \tilde{S}_i 's are as large as our \hat{S}_i 's. Thus, our overhead comparison is reduced to a size comparison between $A(i)$ and $H(i)$. The latter only depends on n while the size of $A(i)$ can be tuned according to the rates α and β . This allows us to reduce the overhead per packet which is impossible for PRABS. Consider the same value n as in Table 2. Whatever the rates are, PRABS's overhead is 10 hashes. If we choose ρ as half the threshold $\frac{\alpha^2}{\beta}$ (as suggested above) then when $(\alpha, \beta) = (0.75, 1.25)$ we get $\frac{1}{\rho} \simeq 5$. In that case, our packet overhead is approximately 5 hashes which is twice less than PRABS's. The closer to $(1, 1)$ our rates are, the smaller our packet overhead becomes. Tuning the packet overhead using the network rates can greatly help regulating the information flow within the communication channel.

5.3 Coding Efficiency

The field used by the MDS code is \mathbb{F}_{2^q} where q is defined as in Section 3. Our results are based on the analysis done in [16]. 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 [15], it was suggested to use Reed-Solomon codes as erasure codes for PRABS. Since Reed-Solomon codes are MDS [20], it is natural to compare their efficiency to our MDS code's one. Karlof *et al.* referred to the original paper by Reed and Solomon [34] to encode/decode. Based on this consideration, the complexity of 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 [34] using the technique developed by Guruswami and Sudan in [13]. This technique which was used in [19] 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, our scheme still gives more benefits than PRABS. 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 [16], 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 Conclusion

In this paper, we introduced a multicast stream authentication scheme which can be considered as an extension of [15, 19]. Our construction ensures non-repudiation of the sender and enables new members to join the communication group at

any block boundary. Contrary to [19], our technique allows recovery of all original data packets. In addition only $O(1)$ signature verifications are performed per block. At the same time, our packet overhead is lower than in [15] 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 constructed a non-asymptotic upper bound on the number of signature verifications to be performed which can be used in practice to deduce an upper bound on the authentication delay our scheme exhibits. We also showed that our erasure code construction was optimal for two reasons. First, MDS codes, by definition, are optimal for correcting erasures. Second, the MDS code construction by Lacan and Fimes was proved in [16] to have a better complexity than most MDS codes used in practice. When compared to Reed-Solomon codes (as used in [19]), 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 scheme. We are aware of the existence of linear time encoding/decoding (non-linear) codes [2, 11, 12, 14, 36]. We did not use them for our protocol because their construction parameters were not flexible enough to fit our network parameters whereas other linear time codes like [18, 21, 38] only provide recovery of all data packets with some probability. In addition, these codes are not MDS which involves extra-generation of symbols to achieve the same capacity of correction.

Acknowledgment

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 and DP0665035. The first author's work was also funded by an iMURS scholarship supported by Macquarie University.

References

- [1] 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.
- [2] 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.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] James Chuan Fu and W. Y. Wendy Lou. *Distribution Theory of Runs and Patterns and its Applications*. World Scientific Publishing, 2003.
- [9] 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.
- [10] Venkatesan Guruswami. *List Decoding of Error-Correcting Codes*. Springer-Verlag, 2004.
- [11] 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.
- [12] Venkatesan Guruswami and Atri Rudra. Explicit capacity-achieving list-decodable codes. Technical Report TR05-133, Electronic Colloquium on Computational Complexity, November 2005.

- [13] 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.
- [14] Piotr Indyk. List-decoding in linear time. Technical Report TR02-024, Electronic Colloquium on Computational Complexity, April 2002.
- [15] 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.
- [16] Jérôme Lacan and Jérôme Fimes. Systematic MDS erasure codes based on Vandermonde matrices. *IEEE Communications Letters*, 8(9):570 – 572, September 2004.
- [17] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and their Applications - Revised Edition*. Cambridge University Press, 2000.
- [18] 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.
- [19] 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.
- [20] Florence Jessiem MacWilliams and Neil James Alexander Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.
- [21] Petar Maymounkov. Online codes. Technical report, New York University, November 2002.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.
- [26] 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.
- [27] 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.
- [28] 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.
- [29] 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.
- [30] Vern Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277 – 292, June 1999.
- [31] 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.
- [32] Adrian Perrig and J. D. Tygar. *Secure Broadcast Communication in Wired and Wireless Networks*. Kluwer Academic Publishers, 2003.
- [33] Josef Pieprzyk, Thomas Hardjono, and Jennifer Seberry. *Fundamentals of Computer Security*. Springer, 2003.
- [34] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of Society for Industrial and Applied Mathematics*, 8(2):300 – 304, June 1960.
- [35] 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.

- [36] 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.
- [37] 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.
- [38] Amin Shokrollahi. Raptor codes. Technical report, Digital Fountain, June 2003.
- [39] Douglas R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 1995.
- [40] 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.
- [41] Chung Kei Wong and Simon S. Lam. Digital signatures for flows and multicasts. *IEEE/ACM Transactions on Networking*, 7(4):502 – 513, August 1999.
- [42] 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.
- [43] Jean-Pierre Zanotti. Le code correcteur C.I.R.C. Available online at: <http://zanotti.univ-tln.fr/enseignement/divers/chapter3.html>.
- [44] Chong zhi Gao and Zheng an 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.

A Study of the cardinality of $\mathbb{F}_{2^{\tilde{m}}}$

In this section, we will study the cardinality of the field used to evaluate/interpolate the polynomial $A(X)$. In order to run Authenticator and Decoder, this field must have at least n elements. That is, we must have: $2^{\tilde{m}} \geq n$. This condition is verified as soon as:

$$\frac{n\mathcal{H} + s}{n\rho + 1} \geq \log_2 n \quad (1)$$

We have: $\rho n \geq 1$. Therefore $\frac{1}{2\rho n}$ is a lower bound of $\frac{1}{n\rho + 1}$. This involves that Inequality (1) is verified as soon as we have:

$$\frac{n\mathcal{H} + s}{2\rho n} \geq \log_2 n$$

It is easy to see that the previous inequality is equivalent to:

$$\frac{\mathcal{H} \ln 2}{2\rho} n - n \ln n + \frac{s \ln 2}{2\rho} \geq 0$$

We introduce the function f defined as:

$$f : \mathbb{R}^{+*} \longrightarrow \mathbb{R} \\ x \longmapsto \frac{\mathcal{H} \ln 2}{2\rho} x - x \ln x + \frac{s \ln 2}{2\rho}$$

This function is differentiable over \mathbb{R}^{+*} and: $\forall x > 0$ $f'(x) = \frac{\mathcal{H} \ln 2}{2\rho} - \ln x - 1$. We obtain:

$$f'(x) \geq 0 \iff x \leq \frac{1}{e} 2^{\frac{\mathcal{H}}{2\rho}}$$

Denote \tilde{x} the real defined as: $\tilde{x} := \frac{1}{e} 2^{\frac{\mathcal{H}}{2\rho}}$. From the previous equivalence, we deduce that f is increasing over $(0, \tilde{x}]$. In addition, we have:

$$\lim_{x \rightarrow 0^+} f(x) = \frac{s \ln 2}{2\rho}$$

Since this limit is positive, we deduce that f is positive over $(0, \tilde{x}]$. Therefore, the restriction of f to $\mathbb{N} \setminus \{0\}$ is positive over $\{1, \dots, [\tilde{x}]\}$. This involves:

$$\forall n \in \{1, \dots, [\tilde{x}]\} \quad 2^{\tilde{m}} \geq n$$

We now have to argue that $\lfloor \tilde{x} \rfloor$ is large enough for real applications. Assume that we have: $\lfloor \tilde{x} \rfloor \leq 2^{20}$. Then, we get:

$$\frac{1}{e} 2^{\frac{\mathcal{H}}{2\rho}} \leq 2^{21}$$

Since e is less than 4 we get: $2^{\frac{\mathcal{H}}{2\rho}} \leq 2^{23}$. Therefore: $\mathcal{H} \leq 46\rho$. Remember that ρ is less than 1. This involves that the length of the digests produced by h does not exceed 46 bits. In this case, h cannot be collision resistant since the birthday attack would require about 2^{23} computations to get a collision with probability at least $\frac{1}{2}$. This contradicts our hypothesis concerning the security of h . Therefore, we must have: $\lfloor \tilde{x} \rfloor > 2^{20}$. This value is sufficient for practical applications since n will be roughly 1000.

B Proof of Theorem 3

We decompose this proof into three parts. In the first one, we will demonstrate the recovery property of our scheme. In the second one, we will prove that $U(n)$ is an upper bound on the number of signature verifications to be performed per block. Finally, we will show that $U(n)$ is $O(1)$ as a function 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.

B.1 Packet Recovery

Since (α, β) is accurate, at least $\lceil \alpha n \rceil$ of the original elements $\{AP_1, \dots, AP_n\}$ are received by the receiver amongst a total of no-more than $\lfloor \beta n \rfloor$ elements. Thus, Step 1 of Decoder is performed successfully. MPR is run on at most $\lfloor \beta n \rfloor$ elements where at least $\lceil \alpha n \rceil$ of them are correct to get back a polynomial of degree at most ρn . Since $\lceil \alpha n \rceil > \sqrt{(\rho n) \lfloor \beta n \rfloor}$ then Poly-Reconstruct can be run and it successfully outputs the list of all polynomials of degree at most ρn which pass through at least $\lceil \alpha n \rceil$ of the given points. In particular, the polynomial $A(X) = a_0 + a_1 X + \dots + a_{\rho n} X^{\rho n}$ belongs to the list generated at Step 2 of MRP. We deduce that $h_1 \parallel \dots \parallel h_n \parallel \sigma \parallel 0^\ell$ belongs to the list output by MRP. This means that at least one signature is verified at Step 2 of Decoder.

If there is another element C' on this list which verifies the signature then C' must end with $\tilde{\ell}$ zeros due to Step 3. It must be written as $C' = h'_1 \parallel \dots \parallel h'_n \parallel \sigma' \parallel 0^{\tilde{\ell}}$ where each h'_i is \mathcal{H} bits long and we must have:

$$\text{Verify}_{\text{PK}}(h(\text{BID} \parallel \alpha \parallel \beta \parallel n \parallel \mathcal{P} \parallel h'_1 \parallel \dots \parallel h'_n), \sigma') = \text{TRUE}$$

The collision resistance property of h involves that the two couples $(h(\text{BID} \parallel \alpha \parallel \beta \parallel n \parallel \mathcal{P} \parallel h_1 \parallel \dots \parallel h_n), \sigma)$ and $(h(\text{BID} \parallel \alpha \parallel \beta \parallel n \parallel \mathcal{P} \parallel h'_1 \parallel \dots \parallel h'_n), \sigma')$ are different. Remember that the key SK is only known to the sender. In addition, the authentication scheme is designed in such a way that the sender only sign one element per value BID. This means that $(h(\text{BID} \parallel \alpha \parallel \beta \parallel n \parallel \mathcal{P} \parallel h'_1 \parallel \dots \parallel h'_n), \sigma')$ is a forgery of the digital signature. This is impossible since we assumed in Section 3 that it was secure.

Therefore, $h_1 \parallel \dots \parallel h_n \parallel \sigma \parallel 0^\ell$ is the only element to verify the signature in the list output by MPR. Thus, the receiver has recovered the n original hashes h_1, \dots, h_n at the end of Step 3. Since h is collision resistant the only elements to be identified at Step 4 are those corresponding to the original received elements. Since (α, β) is accurate there are at least $\lceil \alpha n \rceil$ of them.

Thus, the receiver has at least $\lceil \alpha n \rceil$ of the n modified symbols $\hat{S}_1, \dots, \hat{S}_n$. In addition, the input of DecodeMDS at Step 5 only contains original symbols since the hash function h sorted out the forgeries at Step 4. By construction, the MDS code can correct up to $n - \lceil \alpha n \rceil$ erasures. Therefore, the receiver recovers $S_1, \dots, S_{\lceil \alpha n \rceil}$ at the end of Step 5.

Since Step 6 only consists of removing the pad of length ℓ , we deduce that the receiver has obtained the n original packets P_1, \dots, P_n as output of Decoder.

B.2 Upper Bound for Signature Verification Queries

Since at most one signature verification is performed per element of the list output by MPR, it is sufficient to prove that $U(n)$ is an upper bound on the size of that list. Denote N the number of points on which Poly-Reconstruct is run and T the number of original elements in this list. Due to the accuracy of (α, β) , we have: $T \geq \lceil \alpha n \rceil$ and $N \leq \lfloor \beta n \rfloor$. As noticed before, we have: $T > \sqrt{(\rho n) N}$ 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: $L(N, T) \leq U(n)$.

According to the proof of Proposition 6.15 in Guruswami's thesis [10], 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: $k = \rho n$. Therefore, we obtain:

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

1st bound: We have: $\forall (a, b) \in \mathbb{R}^+ \times \mathbb{R}^+ \quad \sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$. We obtain:

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

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

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

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

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

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

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

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

B.3 Asymptotic Analysis of the Bound

As in [19], 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:

$$\lim_{\substack{n \rightarrow +\infty \\ n \in \mathcal{I}}} \left(\frac{1}{\rho n} \right) = 0 \quad \text{and} \quad \lim_{\substack{n \rightarrow +\infty \\ n \in \mathcal{I}}} \left(\sqrt{\frac{\beta}{\alpha} + \frac{1}{\rho^2 n^2} (1 - \rho\alpha)} \right) = \sqrt{\frac{\beta}{\rho}}$$

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: $U(n) \in O(1)$ which achieves to prove our theorem.