Ensuring Authentication of Digital Information using Cryptographic Accumulators*

Christophe Tartary

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

ctartary@mail.tsinghua.edu.cn

Abstract

In this paper, we study the broadcast authentication problem for both erasure and adversarial networks. Two important concerns for authentication protocols are the authentication delay and the packet overhead. In this paper, we address those points by proposing two schemes based on cryptographic accumulators. Our first scheme is developed for erasure channels and its packet overhead is less than the length of a digest most of the time. This makes our construction one of the least expensive protocols for this network model. Even if the sender processes the stream slightly in delay, the receivers can authenticate packets on-the-fly. Our second scheme is designed for adversarial networks. We show that our packet overhead is less than for the construction by Karlof *et al.* in 2004 and the protocol by Tartary and Wang in 2006 which are two recent efficient schemes dealing with adversarial networks.

Keywords: Stream Authentication, Polynomial Reconstruction, Erasure Channel, Adversarial Channel, Cryptographic Accumulator.

1 Introduction

In this early XXI century, communication networks have expended to such an extent that most human beings are daily connected to them. They are used for many applications such as video-conferences, pay-TV and air traffic control to name a few. A generalized way to distribute information through these networks is broadcasting. However, large-scale broadcasts have the drawback that lost content cannot be retransmitted as the size of the communication group would imply that a single deletion could lead to an overwhelming number of redistribution requests at the sender end. Furthermore, the communication network can be under the influence of malicious users altering the data stream[1]. As a consequence, the security of a broadcasting protocol depends on the properties of the communication network as well as the computational power of the adversaries. In this work, we present authentication protocols secure against computationally bounded opponents.

The goal of streaming is to distribute continuous data such as stock market information. Therefore, the digital content obtained at the receiver end must be authenticated within a short period of delay upon reception. Moreover, many applications transmit private or sensitive information. Thus, non-repudiation of the stream source needs to be provided.

Network bandwidth availability and computational power of end-users are two primary concerns for a stream authentication protocol. Indeed, large packets may create a congestion of the network information flow while receivers with small computational resources will need more time to authenticate data delaying the stream play. Thus, when designing a protocol for stream authentication, one should aim at minimizing both the packet [2] overhead and the computational cost of authenticating information.

The multicast stream authentication problem has been widely studied [6]. Non-repudiation of the sender is provided using a digital signature. However, signing each data packet is not a practical solution as such a cryptographic primitive is generally expensive to generate and/or verify. Thus, a usual approach consists of generating a single signature and amortizing its communication and computation overheads over several packets using hash functions for instance.

*The original version of this paper appears in the proceedings of the 8th International Conference on Cryptology and Network Security (CANS 2009), Lecture Notes in Computer Science, vol. 5888, pp 315 - 333, Springer - Verlag.

[1] In broadcasting, the sequence of information sent into the network is called *stream*.

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

In order to deal with erasures, Perrig *et al*. [26, 27], Challal *et al*. [7], Golle and Modadugu [10] as well as Miner and Staddon [18] appended the hash of each packet to several followers according to specific patterns. They all modeled the packet loss behavior of the network by k-state Markov chains [9] and they obtained bounds on the packet authentication probability. Nevertheless, the drawbacks of these schemes are twofold. First, they are degrading[3] as some received data packets may not be authenticated. Second, they rely on the reception of signed packets which cannot be guaranteed over networks such as the Internet where the User Datagram Protocol only provides a best effort delivery of information. These two issues restrict the range of applications for the previous protocols.

To overcome the issue of signature reliable delivery, a common approach is to split the signature into k smaller parts where only ℓ of them ($\ell < k$) are sufficient to recover it. Signature dispersion can be achieved via various techniques: Park *et al*. [23, 24] as well as Park and Cho [25] used the Information Dispersal Algorithm [28], Al-Ibrahim and Pieprzyk [1] combined linear equations and polynomial interpolation, Pannetrat and Molva [22] utilized erasure codes whereas Desmedt and Jakimoski [8] employed cover-free families [30]. It should be noticed that each of those authentication schemes is non-degrading as well.

The major shortcoming of the previous constructions is that none of them tolerates a single packet injection. This is a central problem when data is distributed over large public networks since it is likely to have some unreliable nodes.

Using an algorithm developed by Guruswami and Sudan called Poly-Reconstruct to solve the polynomial reconstruction problem [11], Lysyanskaya *et al*. [14] constructed a non-degrading authentication protocol exhibiting $O(1)$ signature verification queries per block[4] as a function of the block length n. Their construction was extended by Tartary and Wang [32] who used a Maximal Distance Separable (MDS) code to allow total recovery of all n data packets. In this paper, we denote this latter construction as TWMDS. The augmented packets[5] of TWMDS are $\Omega(\log_2 n)$-bit long as the underlying field used for polynomial operations must have at least n distinct points. Note that the same situation occurs in [14].

Another approach was followed by Karlof *et al*. in [12] when designing PRABS. This protocol combines an erasure code and an accumulator [4] based on a Merkle hash tree [17] to deal with injections. As TWMDS, PRABS only requires $O(1)$ signature verification queries per block. However, its packet overhead is $\Theta(\log_2(n))$-bit long as each augmented packet carries $\lceil \log_2(n) \rceil$ hashes. Nonetheless, the implementations done in [31] tend to infer that, for practical use, PRABS' overhead is larger than TWMDS'.

There exist several cryptographic accumulators. The advantage of using a construction based on hash functions is that aggregation and membership verification are fast contrary to [4, 20]. In [4], checking whether an element was accumulated costs as much as verifying a RSA signature whereas, in [20], it requires two pairing evaluations which is even slower [2, 5].

Nyberg's probabilistic accumulator is also based on hash functions [21]. Recently, Yum *et al*. proposed an improvement allowing to reduce the probability of false membership [35]. In this paper, we present two non-degrading authentication protocols based on this new accumulator, MDS codes and Poly-Reconstruct. Our first scheme is developed for erasure channels. Its overhead is smaller than [22, 23, 24, 25] and it allows each receiver to process information on-the-fly after a short part of the stream has been received. In particular, immediate data authentication can be achieved. Our second protocol is designed for adversarial networks as TWMDS and PRABS. It allows complete recovery of the data stream as TWMDS and we show on implementations that its overhead is smaller than TWMDS' and PRABS' in many situations. Another point worth noting is that our implementations also reinforce the intuition that TWMDS has smaller overhead than PRABS which has only been studied on a particular case so far ($n = 1000$) [31].

This paper is organized as follows. In the next section, we present the mathematical tools needed for the understanding of this paper. In particular, we recall the accumulator construction from [35] which plays a central role in our work. In Section 3, we present our authentication protocol for erasure channels. Our scheme for adversarial networks is studied in Section 4. The last section summarizes our contributions to the broadcast authentication problem.

2 Preliminaries

In this section, we present the network models and erasure correcting codes used in this paper. We also quote the polynomial reconstruction problem which plays an important role for our authentication scheme over adversarial channels. Finally, we recall the cryptographic accumulator construction developed in [35].

[3] An authentication scheme is said to be *non-degrading* if every receiver can authenticate all the data packets he obtained. Otherwise, the scheme is said to be *degrading*.

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

[5] We 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.

2.1 Network Models

We consider that the communication network is under the control of an opponent $\mathcal{O}$.

Erasure Channels. In this model, $\mathcal{O}$ is simply an eavesdropper. Therefore, no injections of malicious packets occur. In other words, any packet collected by the receiver is authentic. We can assume that both sender and receivers have a buffering capacity of n consecutive packets and that at most t packets can be erased over a scope of n elements. This model generalizes the concept of bursts where, in the bursty model, the length of the longest burst occurring in the network is $t = n - 2$ (one packet must been receive on each side of the burst). An illustration is given as Figure 1.

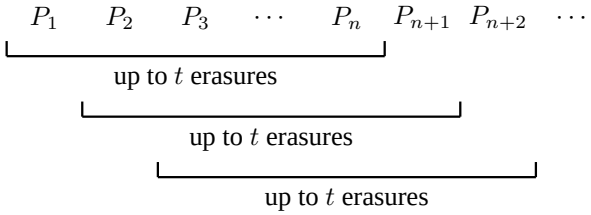


Figure 1: Erasure Channel Model for Streaming

It should be noted that the bursty model has been used to analyze many authentication protocols [10, 18, 26, 27]. This is justified by the work of Yajnik *et al*. [34] who exhibited that the loss pattern of the Internet was bursty in nature. Notice that our model also encompasses [1, 8] as it does not require the t erasures to appear as a burst.

Adversarial Channels. In this case, $\mathcal{O}$ who can drop and rearrange packets of his choice as well as inject bogus data into the network [16]. Without loss of generality, we can assume that a reasonable number of original augmented packets reaches the receivers and not too many incorrect elements are injected by $\mathcal{O}$. We split the data stream into blocks of n packets: $P_1, \ldots, P_n$. In this settings, we introduced two parameters: $\alpha\,(0 < \alpha \leq 1)$ (the *survival* rate) and $\beta\,(\beta \geq 1)$ (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. The use of these two parameters to model $\mathcal{O}$ first appeared in [14] and was subsequently used in [32].

2.2 Correction of Deletions

Since the communication network is a priori unreliable, it is likely that some packets do not reach all the receivers. As in [32], we will use a linear correcting code to overcome this issue. A linear code of length N, dimension K and minimum distance D is denoted $[N, K, D]$.

Theorem 1 ([15]) *Any $[N, K, D]$ code satisfies: $D - 1 \leq N - K$.*

Since any $[N, K, D]$ code can correct up to $D - 1$ erasures [36], such a code can correct at most $N - K$ erasures. To maximize the efficiency of our protocols, we are interested in codes correcting exactly $N - K$ erasures. These codes are called *Maximum Distance Separable* (MDS) codes [15]. TWMDS is also based on this family of codes.

2.3 Reconstructing Polynomials

The Polynomial Reconstruction Problem (PRP) is the following mathematical problem.

Polynomial Reconstruction Problem
Input: Integers D, T and N points $\{(x_i, y_i)\}_{i \in \{1,\ldots,N\}}$ where $x_i, y_i \in F$ for a field F.
Output: All univariate polynomials $P(X) \in F[X]$ of degree at most D such that $y_i = P(x_i)$ for at least T values of $i \in \{1, \ldots, N\}$.

Guruswami and Sudan developed an algorithm called Poly-Reconstruct to solve the PRP [11]. We modify it as in [32] where that new version was denoted MPR. Let $\mathbb{F}_{2^q}$ be the field of the polynomial coefficients. Every element of $\mathbb{F}_{2^q}$ can be represented as a polynomial of degree at most $q - 1$ over $\mathbb{F}_2$. Operations in $\mathbb{F}_{2^q}$ are performed modulo an irreducible polynomial $\mathcal{Q}(X)$ over $\mathbb{F}_2$ having degree q [13]. MPR is represented as Algorithm 1.

Algorithm 1 MPR

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

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

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

2.4 Cryptographic Accumulators

In [35], Yum *et al.* proposed a modified version of Nyberg's cryptographic accumulator [21]. A list $\{x_1, \ldots, x_m\}$ is aggregated into an accumulated value $\mathcal{A}$ using Algorithm 2.

Algorithm 2 ACCUMULATE

Input: Two cryptographic hash functions h and h' outputting $(r\ d)$-bit long and $(k\ \log_2(r))$-bit long digests respectively, a security parameter ϵ, a list of elements to be aggregated $\{x_1, \ldots, x_m\}$.

/* Digests Computation */

1. Compute the digests $h(x_i) := y_{i,1} \| \cdots \| y_{i,r}$ where each $y_{i,j}$ is d-bit long for $i \in \{1, \ldots, m\}$.
2. Compute the digests $h'(x_i) := y'_{i,1} \| \cdots \| y'_{i,r}$ where each $y'_{i,j}$ is $\log_2(r)$-bit long for $i \in \{1, \ldots, m\}$.

/* Binary Strings Generation */

3. For $i \in \{1, \ldots, m\}$, create the string $b_{i,1} \| \cdots \| b_{i,r}$ as follows:
   3.1. Set $b_{i,j} = 1$ for $j \in \{1, \ldots, r\}$.
   3.2. For $\tau \in \{1, \ldots, k\}$, do the following:
      3.2.1. Set: $j = y'_{i,\tau} + 1$.
      3.2.2. Set: $b_{i,j} = 0$ if $\frac{y_{i,j}}{2^d - 1} \leq \epsilon$.

/* Accumulated Value */

4. Compute the binary products: $\forall j \in \{1, \ldots, r\}\ a_j := \prod_{i=1}^{m} b_{i,j}$.

Output: $\mathcal{A} := (a_1, \ldots, a_r)$: accumulated value for the list $\{x_1, \ldots, x_m\}$.

One verifies the membership of an element $\tilde{x}$ to the list $\{x_1, \ldots, x_m\}$ using Algorithm 3.

Algorithm 3 MEMBERSHIP

Input: Two cryptographic hash functions h and h' outputting $(r\ d)$-bit long and $(k\ \log_2(r))$-bit long digests respectively, a security parameter ϵ, the accumulated value $\mathcal{A} = (a_1, \ldots, a_r)$ corresponding to the list $\{x_1, \ldots, x_m\}$ and a candidate element $\tilde{x}$.

/* Digests Computation */

1. Compute the digest $h(\tilde{x}) := \tilde{y}_1 \| \cdots \| \tilde{y}_r$ where each $\tilde{y}_j$ is d-bit long.
2. Compute the digest $h'(\tilde{x}) := \tilde{y}'_1 \| \cdots \| \tilde{y}'_r$ where each $\tilde{y}'_j$ is $\log_2(r)$-bit long.

/* Binary Strings Generation */

3. Create the string $\tilde{b}_1 \| \cdots \| \tilde{b}_r$ as follows:
   3.1. Set $\tilde{b}_j = 1$ for $j \in \{1, \ldots, r\}$.
   3.2. For τ in $\{1, \ldots, n\}$, do the following:
      3.2.1. Set: $j = \tilde{y}'_\tau + 1$.
      3.2.2. Set: $\tilde{b}_j = 0$ if $\frac{\tilde{y}'_j}{2^d - 1} \leq \epsilon$.

/* Accumulated Value */

4. For $j \in \{1, \ldots, r\}$, do the following:
   If ($\tilde{b}_j = 1$ and $a_j = 1$) then Return NO.
5. Return YES.

Output: Decide whether $\tilde{x}$ belongs to $\{x_1, \ldots, x_m\}$.

Yum *et al.* have shown that Algorithm 3 was a YES-bias Monte-Carlo algorithm [29]. They demonstrated that the

value of the bias was:

$$f(\epsilon, k) := \left[1 - \epsilon \left(1 - \frac{\epsilon}{r}\right)^{k\,m}\right]^k$$

based on the *Random Oracle* (RO) model for h and h'.

The issue in [35] is that Yum *et al.* only provide an asymptotic analysis of $f(\epsilon, k)$. Indeed, they substituted $\left(1 - \frac{\epsilon}{r}\right)^{k\,m}$ by $\exp(-\frac{k\,m\,\epsilon}{r})$. However, it is unlikely that a very large number of elements m be accumulated so that this approximation holds.

Fortunately, we can still get some information on how to choose ϵ. Indeed, the partial derivative $\frac{\partial f}{\partial \epsilon}$ is negative. This involves:

$$\forall \epsilon \in [0, 1] \quad f(\epsilon, k) \geq f(1, k)$$

Thus, it is suggested to choose $\epsilon = 1$. In this situation, the bias of the algorithm gets:

$$f(1, k) = \left[1 - \left(1 - \frac{1}{r}\right)^{k\,m}\right]^k$$

As observed in [35], setting $\epsilon = 1$ allows us to completely remove h from the structure of the accumulator. That is, only the cryptographic hash function h' is needed. Given this observation, we assume in the remaining of this paper that $\epsilon = 1$.

Remark 1 *The use of a cryptographic hash function to instantiate the RO model is frequent [33]. In* 2007*, the National Institute of Standards and Technology (NIST) set a competition for a new cryptographic hash algorithm SHA-3 [19]. One of the requirement that the candidates must satisfy is to support pseudo-random functions, in particular, the HMAC construction [3].*

3 Stream Authentication over Erasure Channels

In the remaining of this paper, we work with a unforgeable $\mathcal{S}$-bit long digital signature $(\text{Sign}_{\text{SK}}, \text{Verify}_{\text{PK}})$ [29] the key pair of which (SK,PK) is created by a generator KeyGen and a cryptographic hash function h' outputing $\mathcal{H}'$-bit long digests with $\mathcal{H}' = k \log_2(r)$.

3.1 Authentication Protocol

The stream is a continuous flow of information. First, the sender generates the signature σ on the digest $h'(P_1)$ of the first stream packet. He then encodes the concatenation $\sigma \| h'(P_1)$ using a MDS code of length n and dimension $n - t$. The corresponding codeword is denoted $(C_1 \cdots C_n)$ where each C_i is $\lceil \frac{\mathcal{S}+\mathcal{H}'}{n-t} \rceil$- bit long.

Second, the sender buffers the first n packets $P_1, \ldots, P_n$ as list $\mathcal{L}_1$. He computes the accumulated value $\mathcal{A}_1$ of $\mathcal{L}_1$ and builds the augmented packet: $\text{AP}_1 := 1 \| P_1 \| \mathcal{A}_1 \| C_1$. Third, when a new stream packet P_{n+j-1} ($j \geq 2$) is available, the sender builds the list $\mathcal{L}_j := \{P_j, \ldots, P_{n+j-1}\} \cup \{h'(P_1)\}$. He computes the corresponding accumulated value $\mathcal{A}_j$ and builds the augmented packet: $\text{AP}_j := j \| P_j \| \mathcal{A}_j \| C_{[j]}$ where $[j]$ denotes the unique integer in $\{1, \ldots, n\}$ congruent to j modulo n. In particular: $[n] = [2n] = [3n] = \cdots = n$. We notice that the delay at the sender is n packets as it sends into the network AP_j after P_{n+j-1} be available.

The receiver buffers the first $n - t$ packets $\text{AP}_{r_1}, \ldots, \text{AP}_{r_{n-t}}$ he collects. He can recover the whole codeword $(C_1 \cdots C_n)$ from them and then the signature σ on $h'(P_1)$. This allows to authenticate the $n - t$ accumulated values thanks to $h'(P_1)$ aggregated in them. Those values can in turn be used to authenticate all the received packets. The receiver buffers the accumulated value $\mathcal{A}_{r_{n-t}}$.

When the receiver gets the $(n - t + 1)^{\text{th}}$ packet $\text{AP}_{r_{n-t+1}}$, then it can be authenticated using $\mathcal{A}_{r_{n-t}}$. The receiver buffers $\mathcal{A}_{r_{n-t+1}}$ and the process repeats throughout the stream. One notices that any receiver can verify the authenticity of any packet on-the-fly from the $(n - t + 1)^{\text{th}}$ received packet.

The packet overhead of the scheme is $r + \lceil \frac{\mathcal{S}+\mathcal{H}'}{n-t} \rceil$ bits.

3.2 Analysis of the Protocol

Security. We have the following theorem the proof of which is in Appendix A.

Theorem 2 *Our authentication scheme is a non-degrading authentication protocol. The sender processes data with a delay of n packets throughout streaming while the receiver can authenticate packets on-the-fly from the $(n-t+1)^{\text{th}}$ received element.*

Remark 2 *A single signature is needed to ensure non-repudiation of the **whole** stream.*

Remark 3 *One can notice that, when n is fixed, the lower t is, the larger the delay gets. This might seem to be surprising at first but having low t's implies having small redundancy for the codeword coordinates as $n-t$ is large. That is why one requires more codeword information to reconstruct $(C_1 \cdots C_n)$. The trade-off delay/overhead is an efficiency trade-off.*

Packet Overhead. An important point to notice is that the value $f(1,k)$ does not have any impact on the security of our protocol for erasure channels. Therefore, the only restriction that we have to take into account is $0 < k \leq r$ as this is necessary to construct the accumulator. We minimize the overhead of our construction by tuning the pair (r,k) so that the bit size $\mathcal{H}'$ of the digest output by h' is $k\log_2(r)$. More precisely, we need to choose r as:

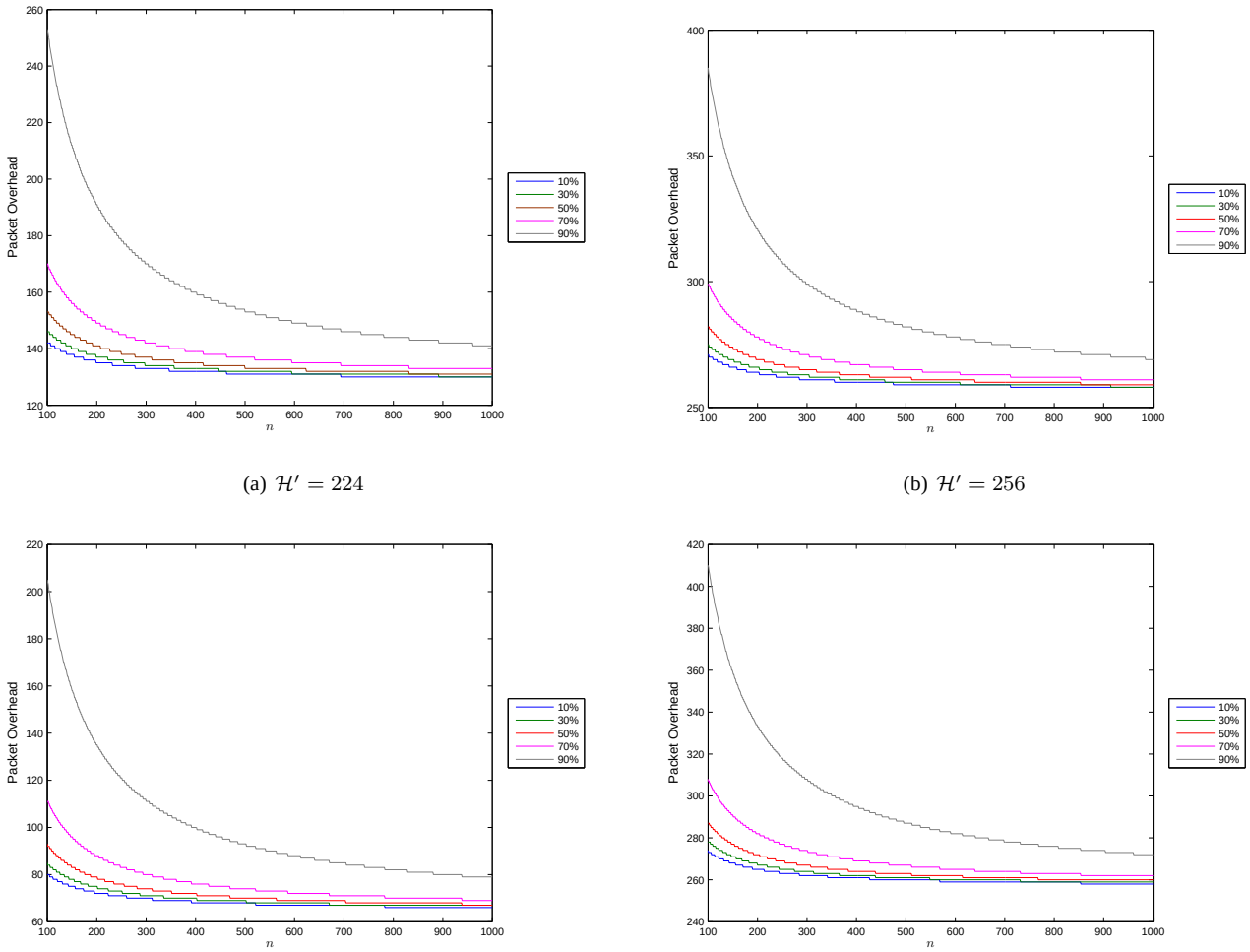
$$r_{\min} := \min\{R \in \mathbb{N} : (0 < K \leq R \text{ and } K \in \mathbb{N} \text{ and } \mathcal{H}' = K\log_2(R))\}$$

In the case of the SHA-3 competition, NIST has required that the new hash function provides message digests of $224, 256, 384$ and 512 bits at least [19]. In this situation, the optimal choice for r is given in Table 1.

$\mathcal{H}'$	224	256	384	512
$r_{\min}$	128	256	64	256

Table 1: Optimal choice for the parameter r.

We plotted the behavior of the packet overhead when the ratio $\frac{t}{n}$ represented $10\%, 30\%, 50\%, 70\%$ and 90% for n varying between 100 and 1000 as Figure 2. We chose to use a 1024-bit long signature to illustrate this result (i.e. $\mathcal{S} = 1024$).



(a) $\mathcal{H}' = 224$

(b) $\mathcal{H}' = 256$

(c) $\mathcal{H}' = 384$

(d) $\mathcal{H}' = 512$

Figure 2: Overhead of our authentication protocol for erasure channels.

We see that in most cases, our packet overhead is less than a digest long. In particular, it is less than in [22, 23, 24, 25].

Remark 4 *Desmedt and Jakimoski's scheme has small overhead as well [8]. Their result is based on optimal choices for cover-free families. The issue is that those optimal families have been shown to exist but they have yet to be constructed as the underlying result by Stinson* et al. *is a non-constructive proof of existence [30].*

4 Stream Authentication over Adversarial Channels

In this channel model, $\mathcal{O}$ can inject bogus data packets into the network. In this situation, we will process the whole data stream per block of n packets $P_1, \ldots, P_n$. Each of these blocks is located within the whole stream using an identification value BID. This approach is used in the different schemes designed for adversarial networks quoted in Section 1 including TWMDS and PRABS. This is to be opposed to the on-the-fly authentication process from the $(n-t+1)^{\text{th}}$ packet at the receiver for our protocol presented in Section 3.1.

4.1 Scheme Overview

Due to erasure of information, we want to generate n augmented packets $\text{AP}_1, \ldots, \text{AP}_n$ such that we can reconstruct all packets $P_1, \ldots, P_n$ from any $\lceil \alpha\, n \rceil$-subset of $\{\text{AP}_1, \ldots, \text{AP}_n\}$. Therefore, our first step consists of encoding $P_1, \ldots, P_n$ using a $[n, \lceil \alpha\, n \rceil, n - \lceil \alpha\, n \rceil + 1]$ code since it can correct up to $n - \lceil \alpha\, n \rceil$ erasures. Note that this approach implies that the elements of the code alphabet are larger than the size of a data packet as the message $(M_1 \cdots M_{\lceil \alpha\, n \rceil})$ to be encoded into the codeword $(C_1 \cdots C_n)$ should represent the concatenation $P_1 \| \cdots \| P_n$.

To ensure non-repudiation of data and to allow new members to join the communication group at any time, we need to generate and distribute a signature which can be reconstructed despite bogus injections by $\mathcal{O}$. Our idea consists of aggregating the n codeword coordinates $C_1, \ldots, C_n$ and signing the corresponding accumulated value $\mathcal{A}$ as σ. We construct a polynomial $A(X)$ of degree at most $\rho\, n$ (for some rational constant ρ), the coefficients of which represent $\mathcal{A} \| \sigma$. We build the augmented packets as:

$$\forall i \in \{1, \ldots, n\}\ \text{AP}_i := \text{BID} \| i \| C_i \| A(i)$$

Upon reception of data, the receiver checks the signature by reconstructing $A(X)$ using MPR. Once the signature σ is verified, the receiver knows the original accumulated value $\mathcal{A}$. Thus, he can identify the correct C_i's amongst the list of elements he got using MEMBERSHIP. According to the definition of α, there must be at least $\lceil \alpha\, n \rceil$ symbols from $C_1, \ldots, C_n$ in his list. Finally, he corrects the erasures using the MDS code and recovers the data packets $P_1, \ldots, P_n$.

4.2 Authentication Protocol

We assume that the values α and β are rational numbers so that we can represent them over a finite number of bits. In order to run Poly-Reconstruct as a subroutine of MPR, we have to choose a parameter $\rho \in (0, \frac{\alpha^2}{\beta})$. Notice that ρ has to be rational since $\rho\, n$ is an integer. Without loss of generality, one can consider that the value ρ is uniquely determined when n, α and β are known. Table 2 summarizes the scheme parameters which are assumed to be publicly known. The bit size $\mathcal{S}$ of the signature and its public key PK are also publicly known. They do not appear in Table 2 as they are considered as general parameters. Note that, once r is known, then k is uniquely determined since the digests of h' are $(k \log_2(r))$-bit long.

n: Block length	A list of irreducible polynomials over $\mathbb{F}_2$
α: Survival rate	β: Flood rate
$\mathcal{P}$: Bit size of data packets	r, k: Parameters of the accumulator hash function h'

Table 2: Public parameters for our authentication scheme over adversarial channels.

The sender of data process the stream as in Algorithm 4. Note that the list of irreducible polynomials is used at Step 1 and Step 3. Furthermore, since any element of $\mathbb{F}_{2^q}^{\sim}$ can be represented as $\lambda_0 Y^0 + \lambda_1 Y_1 + \ldots + \lambda_{\tilde{q}-1} Y^{\tilde{q}-1}$ where each λ_i belongs to $\mathbb{F}_2$, we can define the first n elements as $(0, \ldots, 0)$, $(1, 0, \ldots, 0)$, $(0, 1, 0, \ldots, 0)$, $(1, 1, 0, \ldots, 0)$ and so on until the binary decomposition of $n-1$ (Step 3).

Algorithm 4 AUTHENTICATOR

Input: The secret key SK, the block number BID, Table 2 and n data packets $P_1, \ldots, P_n$.

/* Packet Encoding */

1. Parse $P_1 \| \cdots \| P_n$ as $M_1 \| \cdots \| M_{\lceil \alpha\, n \rceil}$ after padding. Encode the message $(M_1 \cdots M_{\lceil \alpha\, n \rceil})$ into the codeword $(C_1 \cdots C_n)$ using the MDS code over $\mathbb{F}_{2^q}$ with $q = \left\lceil \frac{n\,\mathcal{P}}{\lceil \alpha\, n \rceil} \right\rceil$.

/* Signature Generation and Representation */

2. Compute the accumulated value: $\mathcal{A} = \text{ACCUMULATE}(C_1, \ldots, C_n)$. Construct the block signature as: $\sigma = \text{Sign}_{\text{SK}}(h'(\text{BID} \| n \| \alpha \| \beta \| \mathcal{P} \| r \| \mathcal{A}))$.

3. Denote ξ the smallest element of $\mathbb{N}$ such that:

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

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

/* Construction of Augmented Packets */

4. Build the augmented packets as:

$$\forall i \in \{1, \ldots, n\} \quad \text{AP}_i := \text{BID} \| i \| C_i \| y_i$$

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

Upon reception of data, the receivers use Algorithm 5 to authenticate information.

Algorithm 5 DECODER

Input: The public key PK, the block number BID, Table 2 and the set of received packets RP.

/* Signature Verification */

1. Write the packets as $\text{BID}_i \| j_i \| \widehat{C}_{j_i} \| \widehat{y}_{j_i}$ and discard those having $\text{BID}_i \neq \text{BID}$ or $j_i \notin \{1, \ldots, n\}$. Denote N the number of remaining elements. If $(N < \lceil \alpha\, n \rceil$ or $N > \lfloor \beta\, n \rfloor)$ then the algorithm stops.

2. Rename the remaining elements as $\{\widehat{\text{AP}}_1, \ldots, \widehat{\text{AP}}_N\}$ and write each element as: $\widehat{\text{AP}}_i = \text{BID} \| j_i \| \widehat{C}_{j_i} \| \widehat{y}_{j_i}$ where $j_i \in \{1, \ldots, n\}$. Compute $\tilde{q}$ as in Step 3 of AUTHENTICATOR. Get the irreducible polynomial of degree $\tilde{q}$ from the sender's public list and run MPR on the set $\{(j_i, \widehat{y}_{j_i}), 1 \leq i \leq N\}$ to get a list $\{c_1, \ldots, c_\mu\}$ of candidates for signature verification. If MPR rejects that set then the algorithm stops.

3. Initialize $\widehat{\mathcal{A}} = \emptyset$. While the list has not been exhausted (and the signature not verified yet), pick c_i and write it as: $\mathcal{A}_i \| \sigma_i$ after removing the pad where $\mathcal{A}_i$ is r-bit long. If $\text{Verify}_{\text{PK}}(h'(\text{BID} \| n \| \alpha \| \beta \| \mathcal{P} \| r \| \mathcal{A}_i), \sigma_i) = \text{TRUE}$ then set $\widehat{\mathcal{A}} = \mathcal{A}_i$ and break out the loop. Otherwise, increment i by 1 and start again the While loop.

/* Codeword Reconstruction */

4. If $\mathcal{A}' = \emptyset$ then the algorithm stops. Otherwise, set $C'_k := \emptyset$ for all $k \in \{1, \ldots, n\}$. For each $\widehat{\text{AP}}_i$ written as at Step 2, if $\text{MEMBERSHIP}(h', 1, \widehat{\mathcal{A}}, \widehat{C}_{j_i}) = \text{TRUE}$ then $C'_{j_i} = \widehat{C}_{j_i}$.

5. If $(C'_1 \cdots C'_n)$ has less than $\lceil \alpha\, n \rceil$ non-empty symbols then the algorithm stops. Otherwise, denote it into the message $(M'_1 \cdots M'_{\lceil \alpha\, n \rceil})$.

/* Packet Recovery */

6. If the decoding fails then the algorithm stops. Otherwise, remove the pad from $M'_1 \| \cdots \| M'_{\lceil \alpha\, n \rceil}$ and write the remaining string as $P'_1 \| \cdots \| P'_n$ where each P'_i is $\mathcal{P}$ bits long.

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

4.3 Analysis of the Protocol

Security. As the channel model allows an adversary to inject bogus elements into the network, we adopt the same security definition as in [32].

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

*i) A key pair (*SK,PK*) is generated by* KeyGen*.*

ii) $\mathcal{O}$ is given: (a) The public key PK *and (b) Oracle access to* AUTHENTICATOR *(but $\mathcal{O}$ can only issue at most one query with the same block identification tag* BID*).*

iii) $\mathcal{O}$ outputs $(\text{BID}, n, \alpha, \beta, \mathcal{P}, r, \text{RP})$.

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

a) (violation of the correctness property) $\mathcal{O}$ succeeds to output RP *such that even if it contains $\lceil \alpha\, n \rceil$ packets (amongst a total not exceeding $\lfloor \beta\, n \rfloor$ elements) of some authenticated packet set* AP_i *for block identification tag* BID *and parameters $n, \alpha, \beta, \mathcal{P}$, the decoder fails to authenticate all the correct packets.*

b) (violation of the security property) $\mathcal{O}$ succeeds to output RP *such that the decoder outputs $\{P'_1, \ldots, P'_n\}$ that was never authenticated by* AUTHENTICATOR *for the value* BID *and parameters $n, \alpha, \beta, \mathcal{P}$.*

Remark 5 *A protocol with is secure and (α, β)-correct is non-degrading.*

We have the following theorem the proof of which can be found as Appendix B.

Theorem 3 *If our authentication scheme is either insecure or not $(\alpha, \beta)-$correct, then one can create a genuine element passing successfully* MEMBERSHIP.

Packet Overhead. Due to Theorem 3, we have to choose r in order to reduce the value of the YES-bias $f(1, k)$ as much as possible to ensure the security of the authentication protocol.

Since the packet overhead is:

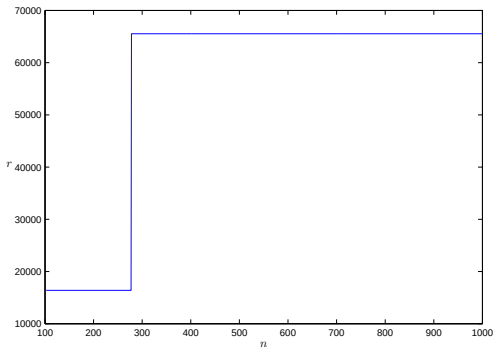
$$\left(\frac{n\,\mathcal{P}}{\lceil \alpha\, n \rceil} - \mathcal{P} \right) + \left\lceil \frac{r + \mathcal{S} + \xi}{\rho\, n + 1} \right\rceil \text{ bits}$$

we look for the smallest value of r such that:

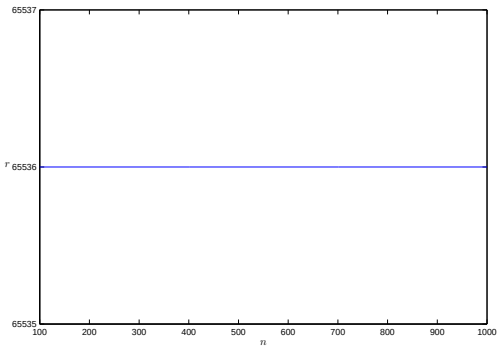
$$f(1, k) \leq t_{\text{bias}}$$

where t_{bias} is the threshold value for the bias. Note that this minimal value for r is independent from both rates α and β.

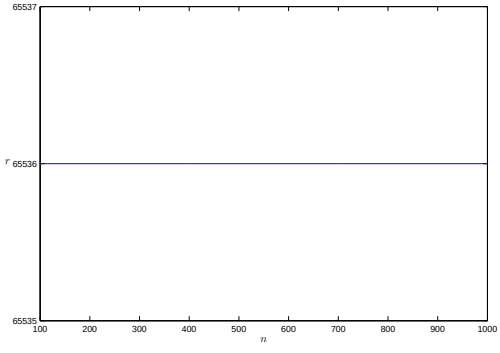
Figure 3 represents the minimal values of r for n between 100 and 1000 when $t_{\text{bias}} = 10^{-10}$. As in Section 3.2, we used four digest length sizes for h': $224, 256, 384, 512$.



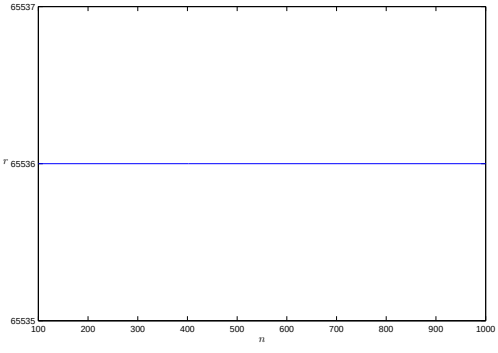
(a) $\mathcal{H}' = 224$



(b) $\mathcal{H}' = 256$



(c) $\mathcal{H}' = 384$



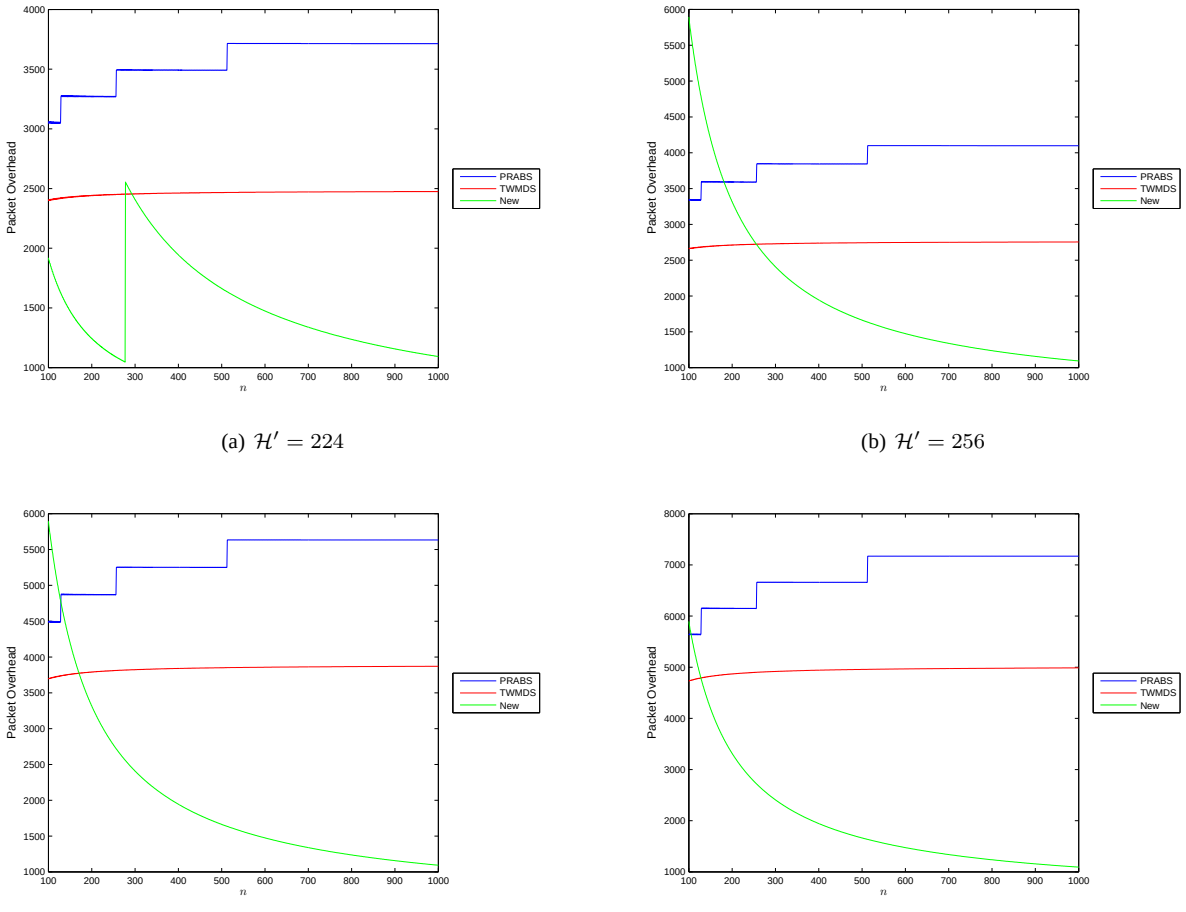
(d) $\mathcal{H}' = 512$

Figure 3: Overhead of our authentication protocol for erasure channels.

In order to provide a fair comparison with PRABS, we have to slightly modify Karlof *et al.*'s construction so that it also allows recovery of the whole data stream as TWMDS and our construction. This is at the cost of using a MDS code which leads to the additional overhead for PRABS of $\frac{n\,\mathcal{P}}{\lceil \alpha\, n \rceil} - \mathcal{P}$ bits. Therefore, in our implementations, the packet overhead for PRABS becomes:

$$\left(\frac{n\,\mathcal{P}}{\lceil \alpha\, n \rceil} - \mathcal{P} \right) + \mathcal{H}' \lceil \log_2(n) \rceil \text{ bits}$$

We performed comparisons for $\alpha \in \{0.5, 0.75, 0.8, 0.9\}$ and $\beta = \{1.1, 1.25, 1.5, 2\}$ for the four digest sizes $\mathcal{H}' \in \{224, 256, 384, 512\}$. We chose $\rho = \frac{\alpha}{2\,\beta^2}$ as suggested in [32] . The results of our implementations are depicted from Figure 4.



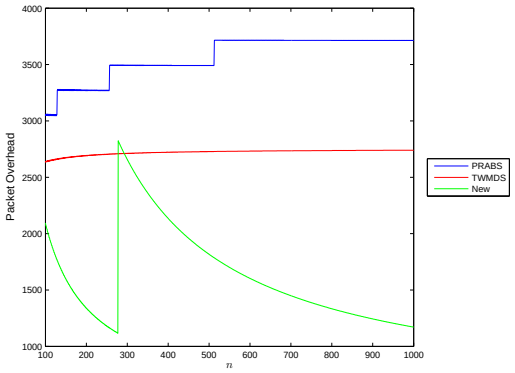
(a) $\mathcal{H}' = 224$

(b) $\mathcal{H}' = 256$

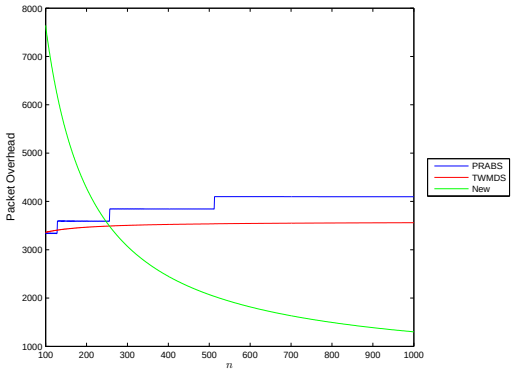
(c) $\mathcal{H}' = 384$

(d) $\mathcal{H}' = 512$

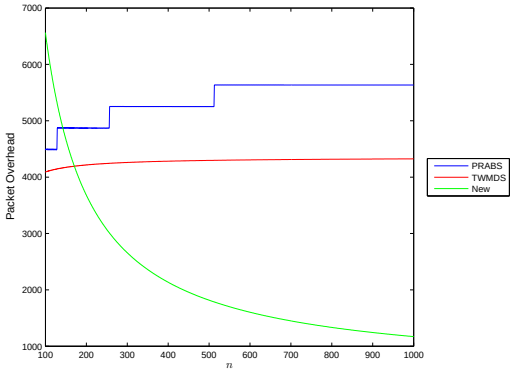
Figure 4: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.5$ and $\beta = 1.1$.



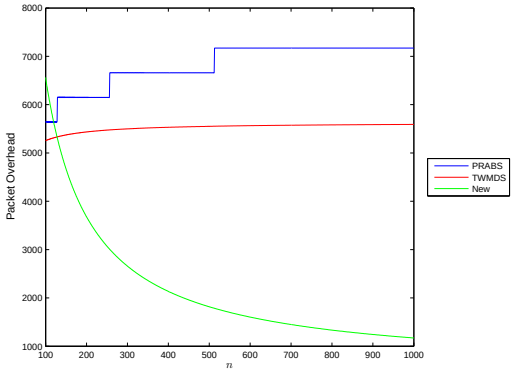
(a) $\mathcal{H}' = 224$



(b) $\mathcal{H}' = 256$

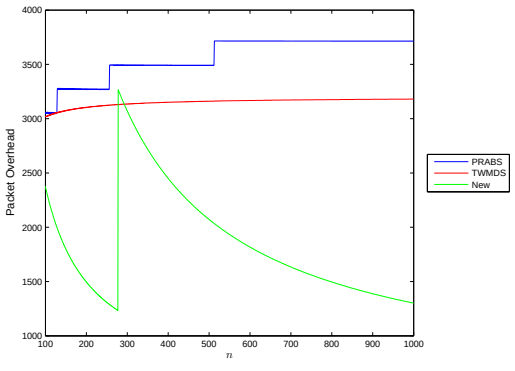


(c) $\mathcal{H}' = 384$

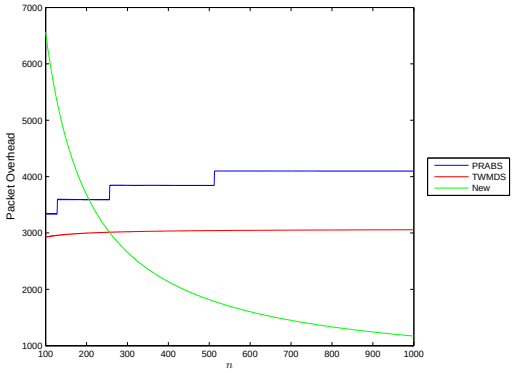


(d) $\mathcal{H}' = 512$

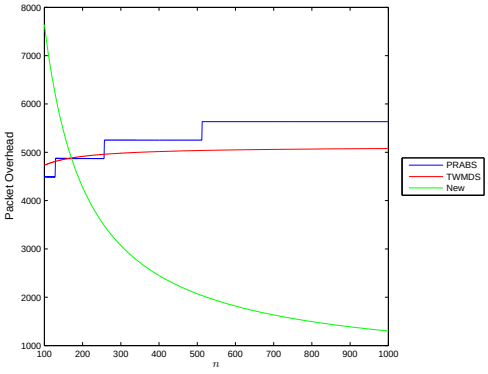
Figure 5: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.5$ and $\beta = 1.25$.



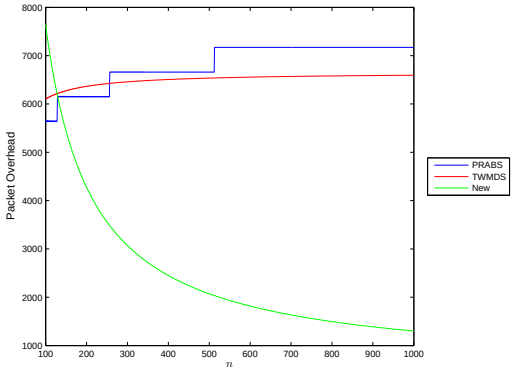
(a) $\mathcal{H}' = 224$



(b) $\mathcal{H}' = 256$

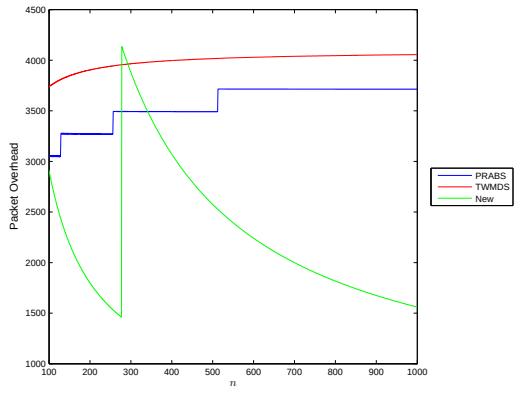


(c) $\mathcal{H}' = 384$

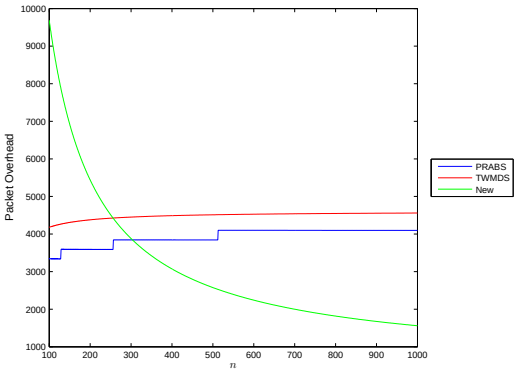


(d) $\mathcal{H}' = 512$

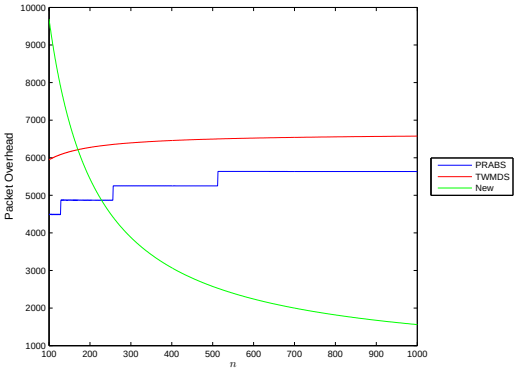
Figure 6: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.5$ and $\beta = 1.5$.



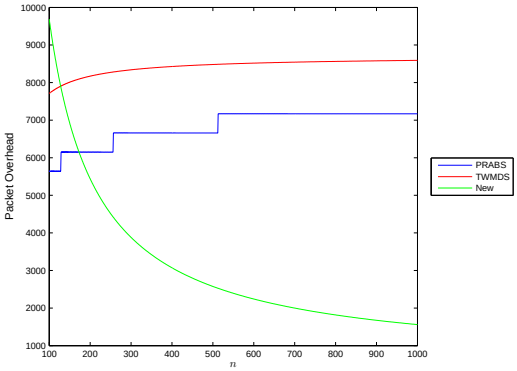
(a) $\mathcal{H}' = 224$



(b) $\mathcal{H}' = 256$

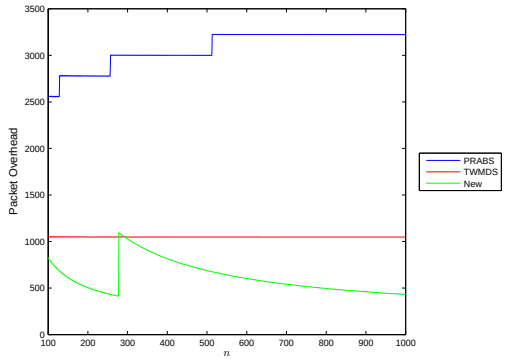


(c) $\mathcal{H}' = 384$

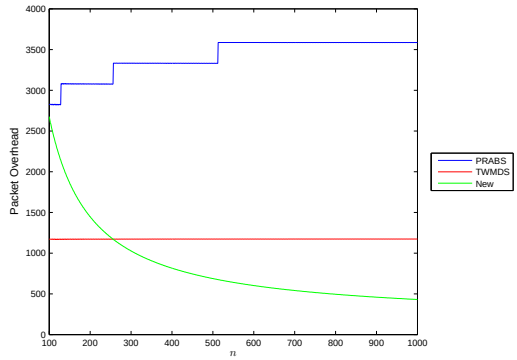


(d) $\mathcal{H}' = 512$

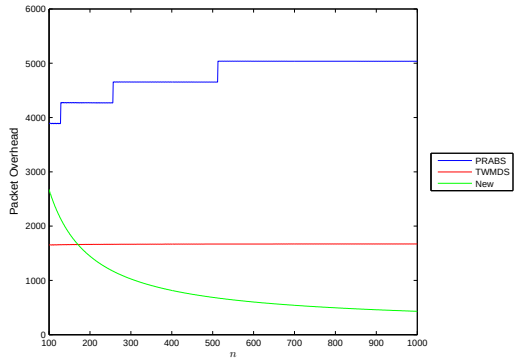
Figure 7: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.5$ and $\beta = 2$.



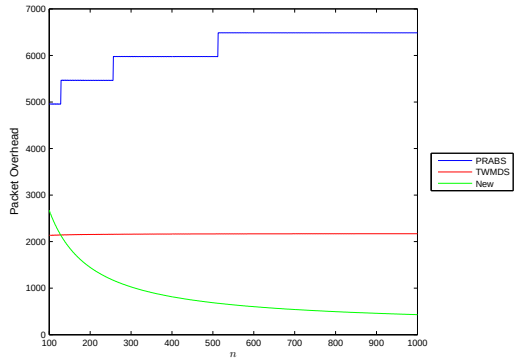
(a) $\mathcal{H}' = 224$



(b) $\mathcal{H}' = 256$

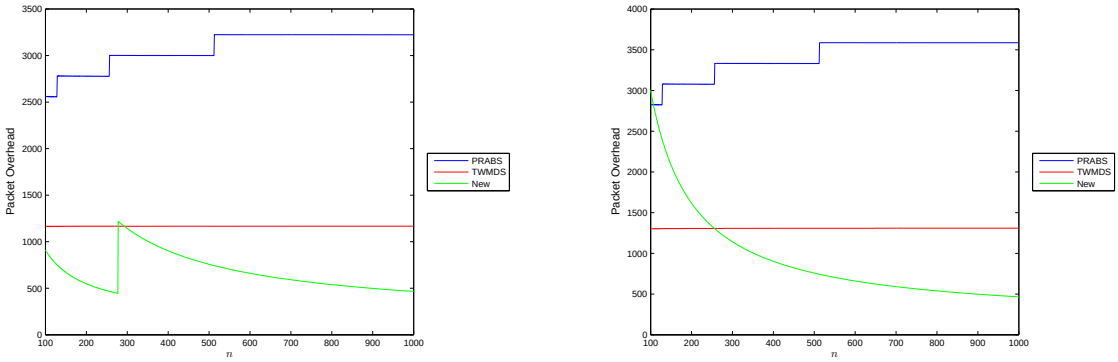


(c) $\mathcal{H}' = 384$

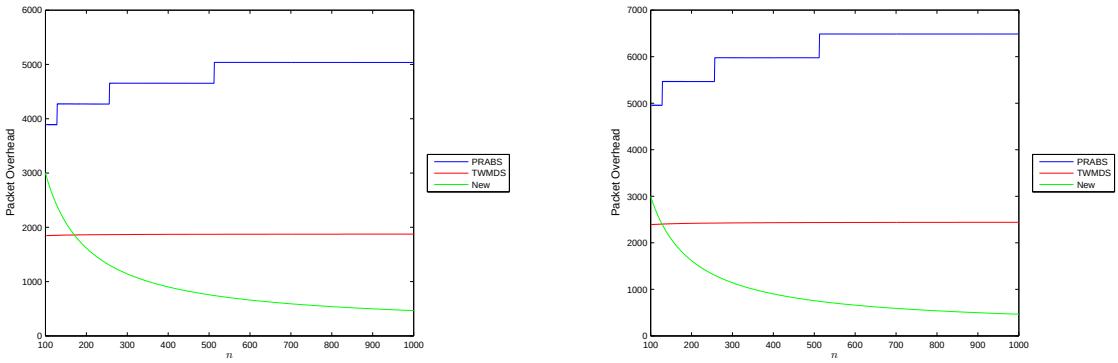


(d) $\mathcal{H}' = 512$

Figure 8: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.75$ and $\beta = 1.1$.

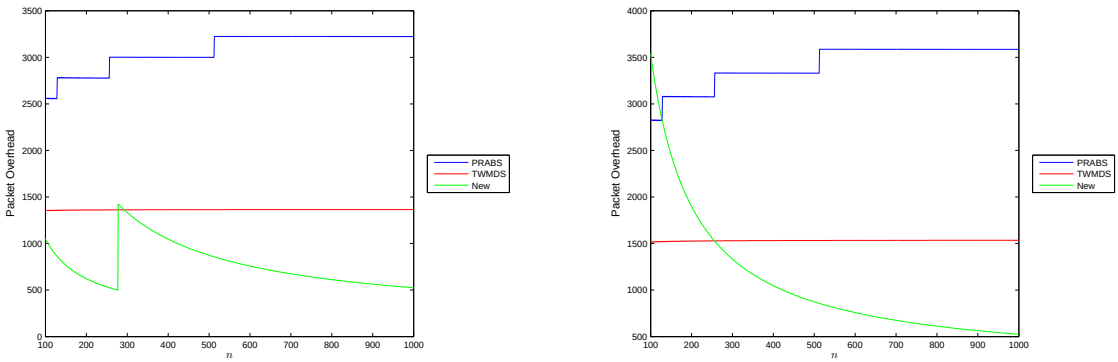


(a) $\mathcal{H}' = 224$ (b) $\mathcal{H}' = 256$

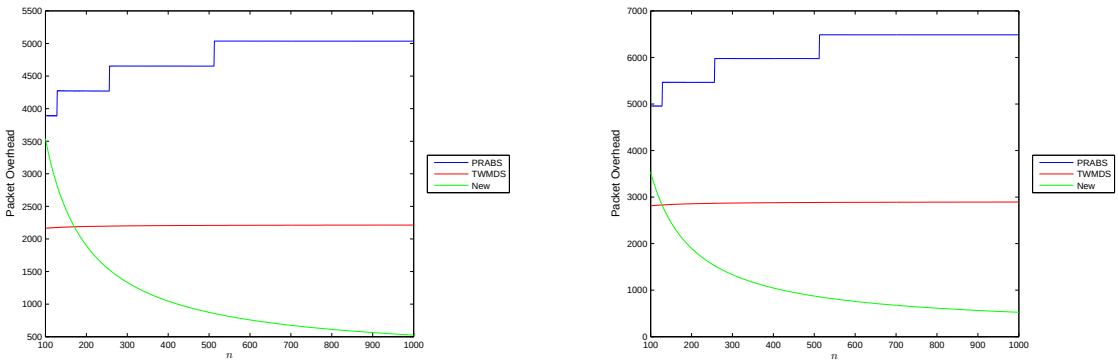


(c) $\mathcal{H}' = 384$ (d) $\mathcal{H}' = 512$

Figure 9: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.75$ and $\beta = 1.25$.

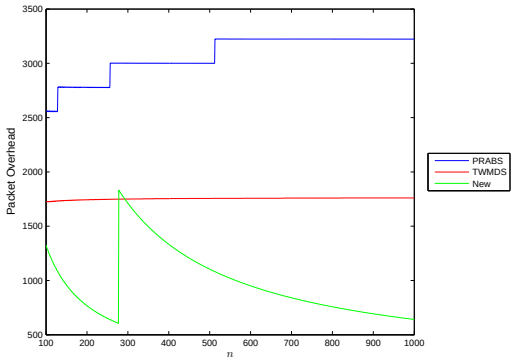


(a) $\mathcal{H}' = 224$ (b) $\mathcal{H}' = 256$

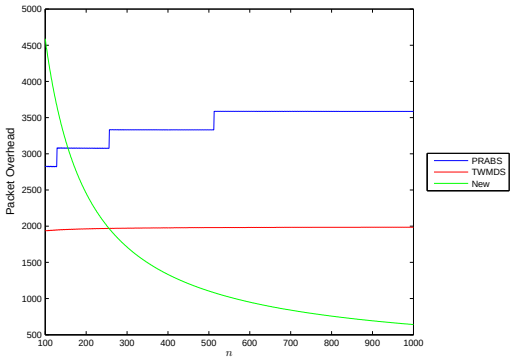


(c) $\mathcal{H}' = 384$ (d) $\mathcal{H}' = 512$

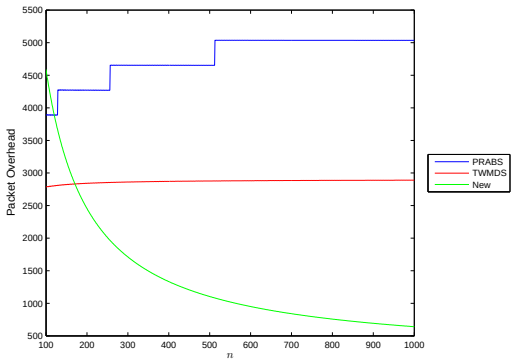
Figure 10: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.75$ and $\beta = 1.5$.



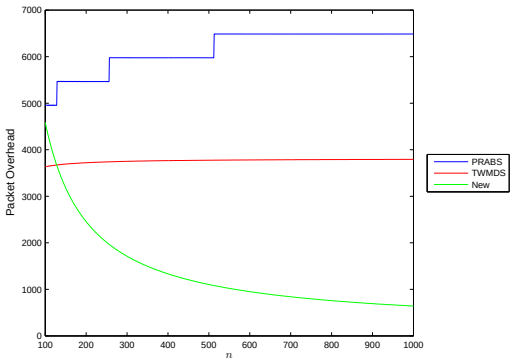
(a) $\mathcal{H}' = 224$



(b) $\mathcal{H}' = 256$

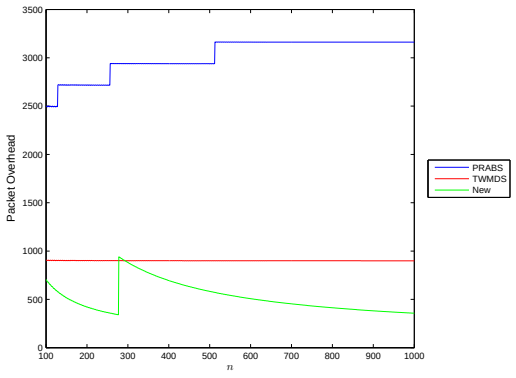


(c) $\mathcal{H}' = 384$

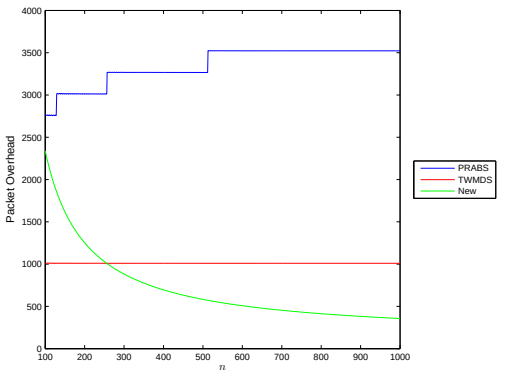


(d) $\mathcal{H}' = 512$

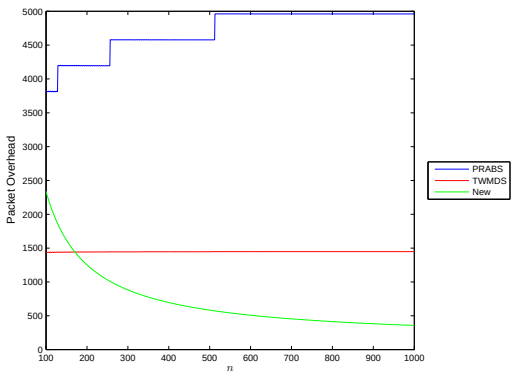
Figure 11: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.75$ and $\beta = 2$.



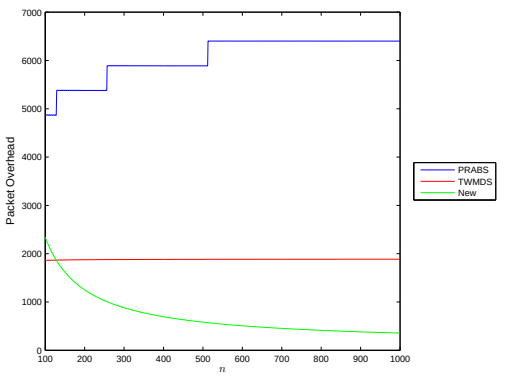
(a) $\mathcal{H}' = 224$



(b) $\mathcal{H}' = 256$

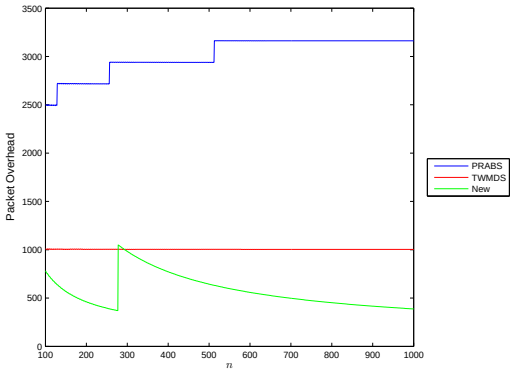


(c) $\mathcal{H}' = 384$

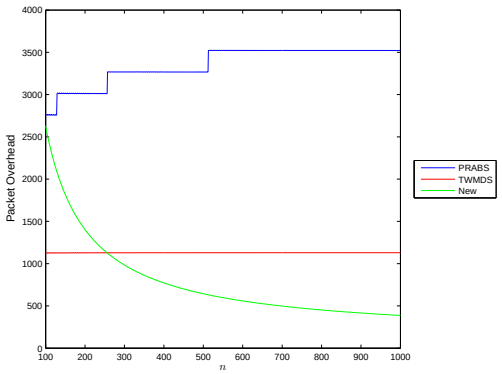


(d) $\mathcal{H}' = 512$

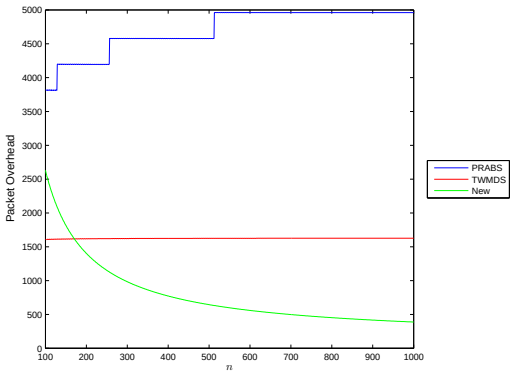
Figure 12: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.8$ and $\beta = 1.1$.



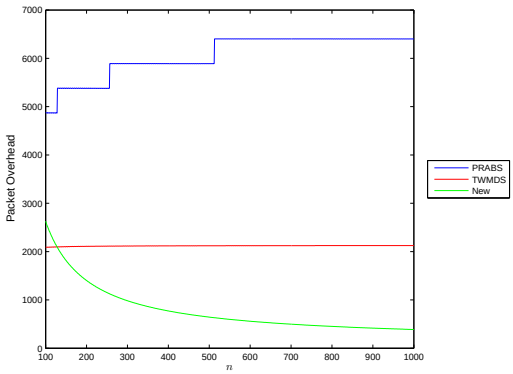
(a) $\mathcal{H}' = 224$



(b) $\mathcal{H}' = 256$

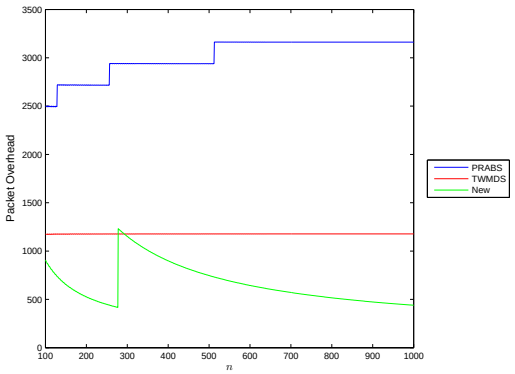


(c) $\mathcal{H}' = 384$

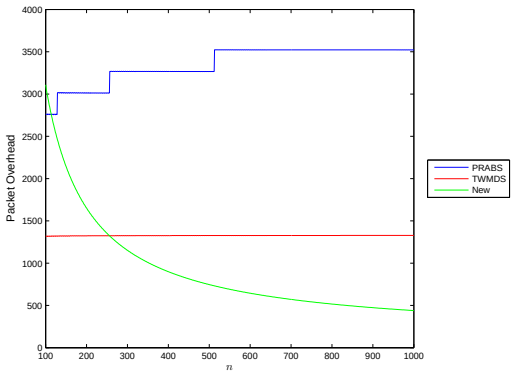


(d) $\mathcal{H}' = 512$

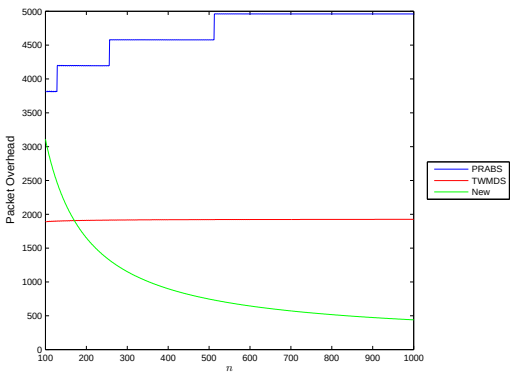
Figure 13: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.8$ and $\beta = 1.25$.



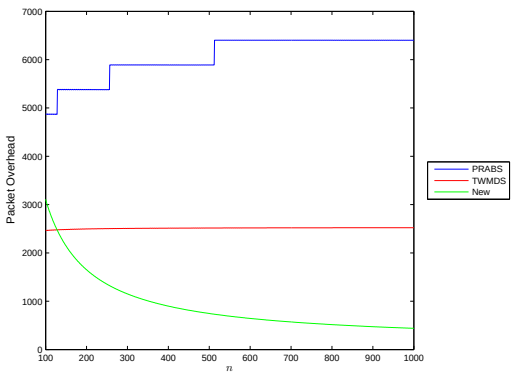
(a) $\mathcal{H}' = 224$



(b) $\mathcal{H}' = 256$

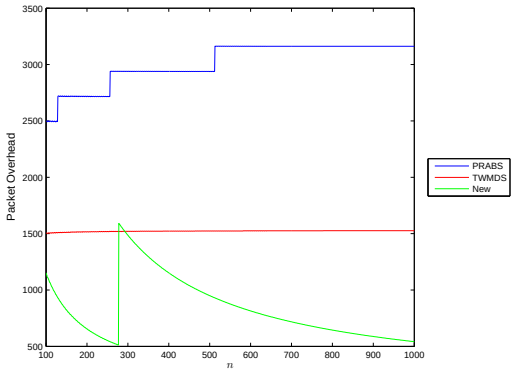


(c) $\mathcal{H}' = 384$

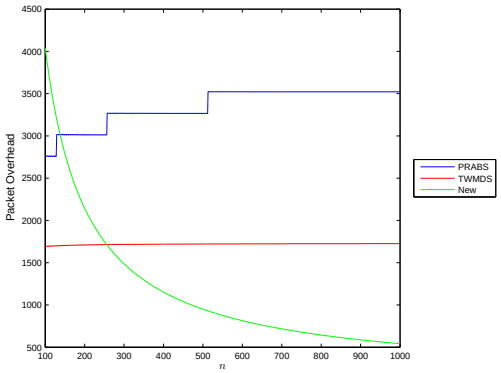


(d) $\mathcal{H}' = 512$

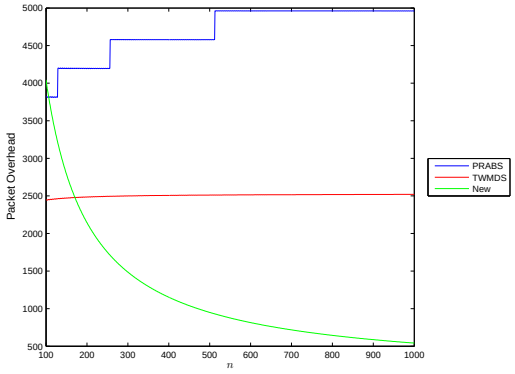
Figure 14: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.8$ and $\beta = 1.5$.



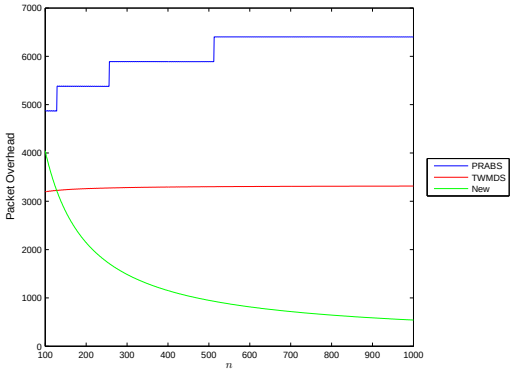
(a) $\mathcal{H}' = 224$



(b) $\mathcal{H}' = 256$

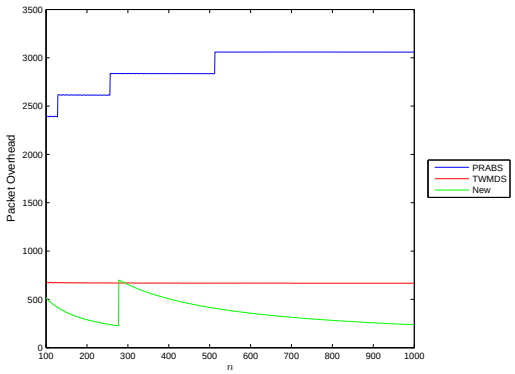


(c) $\mathcal{H}' = 384$

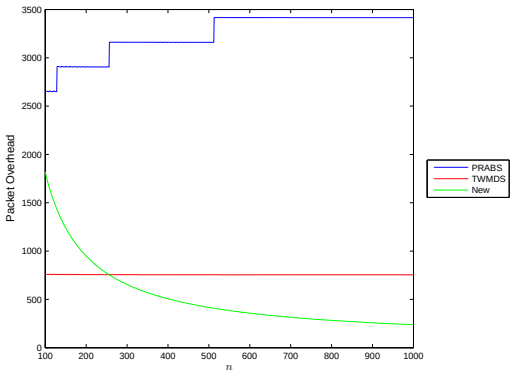


(d) $\mathcal{H}' = 512$

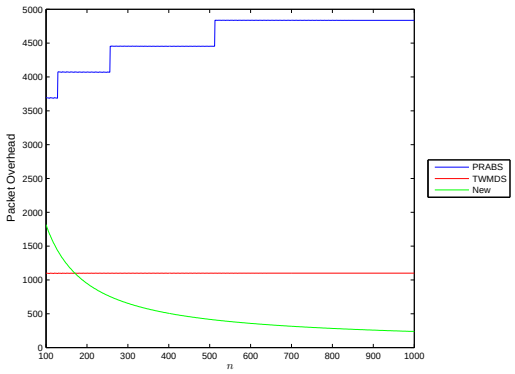
Figure 15: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.8$ and $\beta = 2$.



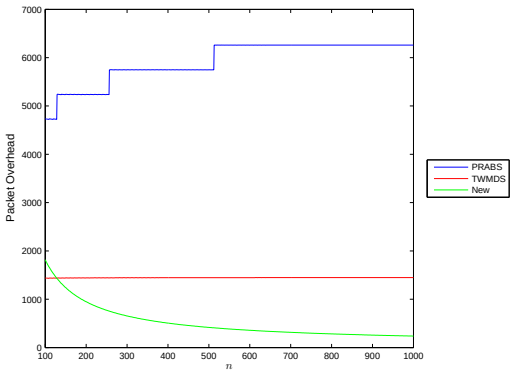
(a) $\mathcal{H}' = 224$



(b) $\mathcal{H}' = 256$

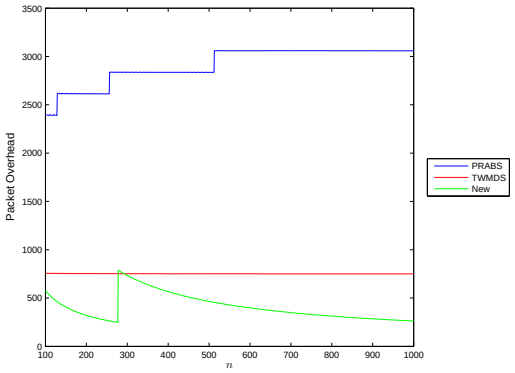


(c) $\mathcal{H}' = 384$

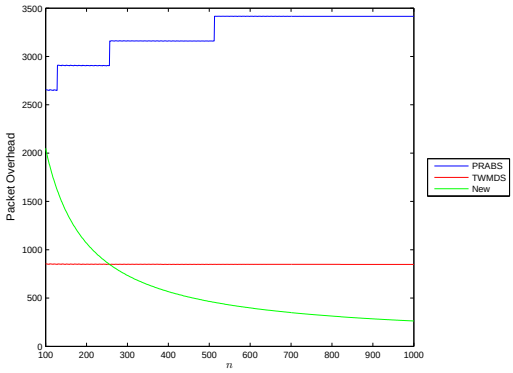


(d) $\mathcal{H}' = 512$

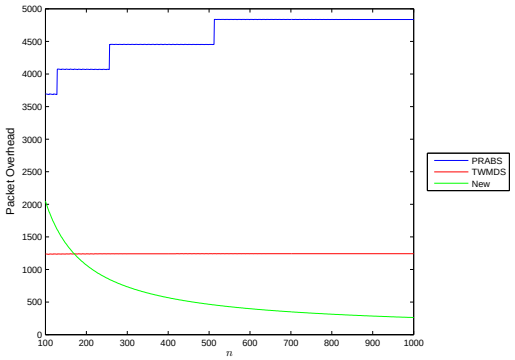
Figure 16: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.9$ and $\beta = 1.1$.



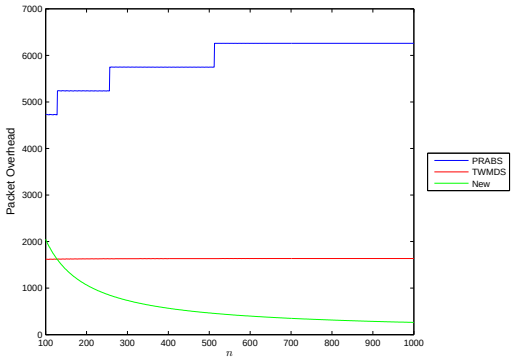
(a) $\mathcal{H}' = 224$



(b) $\mathcal{H}' = 256$

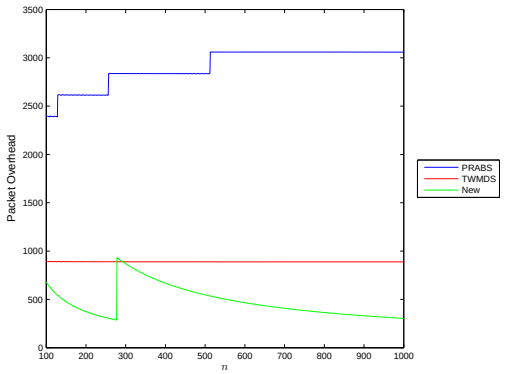


(c) $\mathcal{H}' = 384$

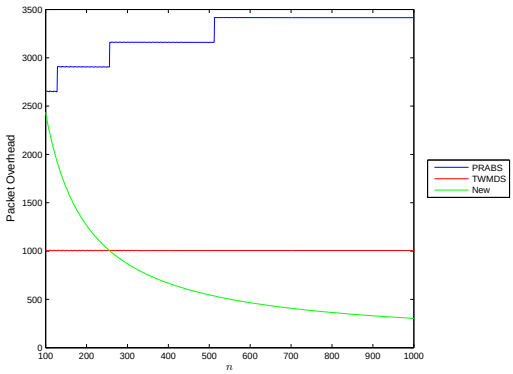


(d) $\mathcal{H}' = 512$

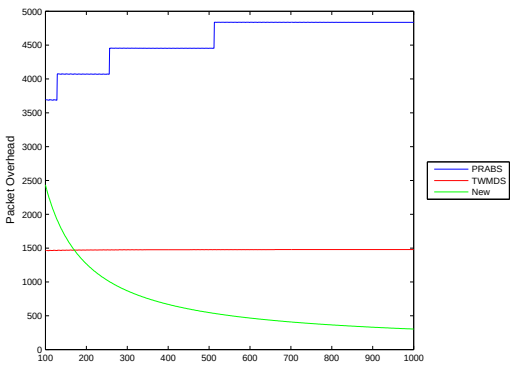
Figure 17: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.9$ and $\beta = 1.25$.



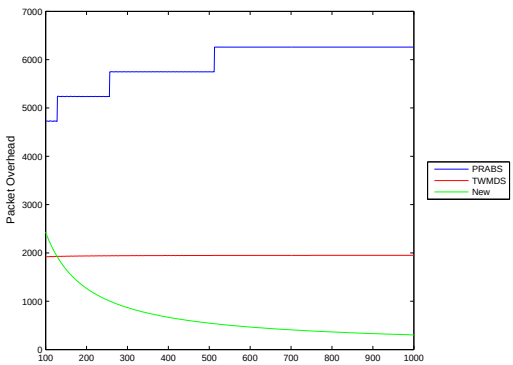
(a) $\mathcal{H}' = 224$



(b) $\mathcal{H}' = 256$

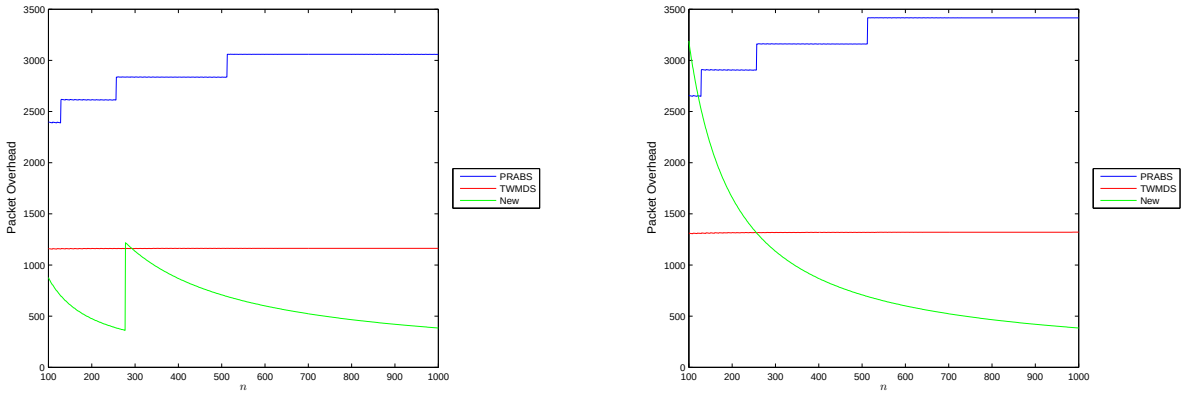


(c) $\mathcal{H}' = 384$

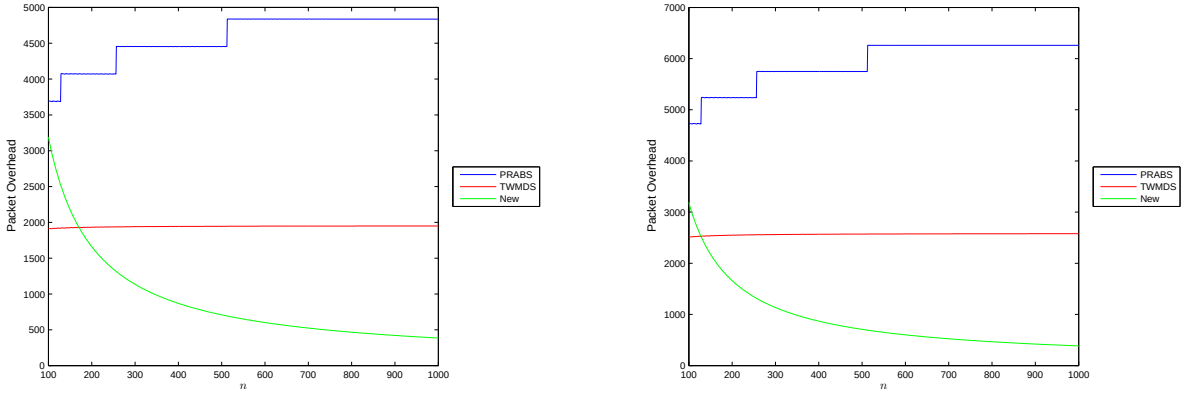


(d) $\mathcal{H}' = 512$

Figure 18: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.9$ and $\beta = 1.5$.



(a) $\mathcal{H}' = 224$ (b) $\mathcal{H}' = 256$



(c) $\mathcal{H}' = 384$ (d) $\mathcal{H}' = 512$

Figure 19: Overhead comparison between PRABS, TWMDS and our scheme when $\alpha = 0.9$ and $\beta = 2$.

Our results show that, in most situations, the overhead of our new scheme is much smaller than PRABS' and TWMDS'. This is particularly acute when n gets large. Furthermore, those implementations demonstrate that the overhead of TWMDS is smaller than PRABS'. This extends the comparative survey between PRABS and TWMDS done so far which was only focused on the case $n = 1000$ [31, 32].

5 Conclusion

In this paper, we presented two protocols for the broadcast authentication problem using a modified version of Nyberg's accumulator due to Yum *et al*. Our first scheme was related to erasure channels. We showed that its packet overhead was less than the length of a digest and, in particular, far less than [22, 23, 24, 25]. Even if the sender processes the stream in delay of n packets, the receivers can authenticate packets on-the-fly from the $(n - t + 1)^{\text{th}}$ received element to the end of the stream (if any). In addition, a single signature is needed to provide the non-repudiation of the whole stream. Our second scheme was designed for adversarial networks. It is obvious that the number of signature queries at the receiver is the same as for TWMDS due to the use of Poly-Reconstruct in both constructions. This number turns to be $O(1)$ as a function of the block length n [32]. Furthermore, the packet overhead of our new scheme is smaller than PRABS' and TWMDS'. Another interesting result from this comparative study was that we obtained more a extensive comparison between PRABS and TWMDS showing that the overhead of TWMDS was smaller.

Acknowledgments

The author is grateful to the anonymous reviewers for their comments to improve the quality of this paper. This work was supported by the National Natural Science Foundation of China grant 60553001 and the National Basic Research Program of China grants 2007CB807900 and 2007CB807901.

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] Paulo S.L.M. Barreto, Hae Y. Kim, Ben Lynn, and Michael Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology - Crypto '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 354 – 369, Santa Barbara, USA, August 2002. Springer - Verlag.

[3] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Message authentication using hash functions - the HMAC construction. In *RSA Laboratories' CryptoBytes*, volume 2, Spring 1996.

[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] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297 – 319, September 2004.

[6] 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.

[7] Yacine Challal, Abdelmadjid Bouabdallah, and Hatem Bettahar. H_2A: Hybrid hash-chaining scheme for adaptive multicast source authentication of media-streaming. *Computer & Security*, 24(1):57 – 68, February 2005.

[8] Yvo Desmedt and Goce Jakimoski. Non-degrading erasure-tolerant information authentication with an application to multicast stream authentication over lossy channels. In *Topics in Cryptology - CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 324 – 338, San Francisco, USA, February 2007. Springer - Verlag.

[9] James Chuan Fu and W. Y. Wendy Lou. *Distribution Theory of Runs and Patterns and its Applications*. World Scientific Publishing, 2003.

[10] Phillippe Golle and Nagendra Modadugu. Authenticating streamed data in the presence of random packet loss. In *Symposium on Network and Distributed Systems Security*, pages 13 – 22, San Diego, USA, February 2001. Internet Society.

[11] 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.

[12] 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*, San Diego, USA, February 2004.

[13] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and their Applications (Revised Edition)*. Cambridge University Press, 2000.

[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] Florence Jessiem MacWilliams and Neil James Alexander Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[16] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[17] 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.

[18] 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.

[19] National Institute of Standards and Technology. Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. *Federal Register*, 72(212):62212 – 62220, November 2007.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] Adrian Perrig and J. D. Tygar. *Secure Broadcast Communication in Wired and Wireless Networks*. Kluwer Academic Publishers, 2003.

[28] Michael O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the Association for Computing Machinery*, 36(2):335 – 348, April 1989.

[29] Douglas R. Stinson. *Cryptography: Theory and Practice (Third Edition)*. Discrete Mathematics and Its Applications. Chapman & Hall/CRC, 2006.

[30] Douglas R. Stinson, Ruizhong Wei, and L. Zhu. Some new bounds for cover-free families. *Journal of Combinatorial Theory, Series A*, 90(1):224 – 234, April 2000.

[31] Christophe Tartary. *Authentication for Multicast Communication*. PhD thesis, Department of Computing - Macquarie University, October 2007.

[32] Christophe Tartary and Huaxiong Wang. Achieving multicast stream authentication using MDS codes. In *5th International Conference on Cryptology and Network Security*, volume 4301 of *Lecture Notes in Computer Science*, pages 108 – 125, Suzhou, China, December 2006. Springer - Verlag.

[33] Henk C. A. van Tilborg. *Encyclopedia of Cryptography and Security*. Springer, 2005.

[34] Maya Yajnik, Sue Moon, Jim Kurose, and Don Towsley. Measurement and modeling of the temporal dependence in packet loss. In *IEEE INFOCOM 1999*, volume 1, pages 345 – 352, New York, USA, March 1999. IEEE Press.

[35] Dae Hyun Yum, Jae Woo Seo, and Pil Joong Lee. Generalized combinatoric accumulator. *IEICE Transactions on Information and Systems*, E91-D(5):1489 – 1491, May 2008.

[36] Jean-Pierre Zanotti. Le code correcteur C.I.R.C. Available online at: `http://zanotti.univ-tln.fr/enseignement/divers/chapter3.html`.

A Proof of Theorem 2

Denote $\text{AP}_{r_1}, \ldots, \text{AP}_{r_{n-t}}$, the first $n-t$ augmented packets obtained by the receiver. Due to our channel model, we have $r_{n-t} - r_1 < n$.

As a consequence, $[r_1], \ldots, [r_{n-t}]$ are $n-t$ distinct values from $\{1, \ldots, n\}$. Thus, the receiver can uniquely identify $C_{[r_1]}, \ldots, C_{[r_{n-t}]}$ to their $n-t$ corresponding values from $C_1, \ldots, C_n$ by using the mapping $x \mapsto [x]$ over the values $r_1, \ldots, r_{n-t}$ contained within $\text{AP}_{r_1}, \ldots, \text{AP}_{r_{n-t}}$.

Using the correction capacity of the MDS code, the receiver can recover the codeword $(C_1 \cdots C_n)$ and then its corresponding message $(M_1 \cdots M_{n-t})$. This message easily leads to $\sigma \| h'(P_1)$ as this string represents the first $\mathcal{S} + \mathcal{H}'$ bits of $M_1 \| \cdots \| M_{n-t}$. Finally, the receiver verifies the authenticity of the signature σ on $h'(P_1)$ using $\text{Verify}_{\text{PK}}$.

So far, the receiver only authenticated $h'(P_1)$. It should be noticed that, since we are working over an erasure channel, it is sufficient to show that $h'(P_1)$ has been accumulated within the value $\mathcal{A}_{r_1}$ to authenticate this value and therefore every single element aggregated within (using MEMBERSHIP). We have two cases to consider.

1. $r_1 = 1$: The receiver obtained the first augmented AP_1 of the stream. In this case, he can verify the authenticity of AP_1 by computing $h'(P_1)$. He authenticates $\mathcal{A}_1$ using $h'(P_1)$ and MEMBERSHIP. The remaining $n-t-1$ augmented packets $\text{AP}_{r_2}, \ldots, \text{AP}_{r_{n-t}}$ belong to $\{\text{AP}_1, \ldots, \text{AP}_n\}$. Since $\mathcal{L}_1 = \{P_1, \ldots, P_n\}$, the validity of $P_{r_2}, \ldots, P_{r_{n-t}}$ can be checked using MEMBERSHIP and $\mathcal{A}_1$.

2. $r_1 \geq 2$: The receiver did not receive AP_1. This case can also be seen as the receiver joining the communication group after the beginning of streaming.

Since the receiver authenticated $h'(P_1)$ thanks to the digital signature, he can check the authenticity of $\mathcal{A}_{r_1}$ using $h'(P_1)$ and MEMBERSHIP.

The remaining $n-t-1$ augmented packets $\text{AP}_{r_2}, \ldots \text{AP}_{r_{n-t}}$ can be authenticated using $\mathcal{A}_{r_1}$ and MEMBERSHIP since $P_{r_2}, \ldots, P_{r_{n-t}}$ have been aggregated into $\mathcal{A}_{r_1}$ since: $r_1 < r_2 < \cdots < r_{n-t} \leq r_1 + (n-1)$.

Up to this point, we showed that the receiver could authenticate the first $n-t$ augmented packets he got. In order to terminate the demonstration of this theorem, it remains to prove that the receiver can authenticate (on-the-fly) all the following augmented packets he obtains: $\text{AP}_{r_{n-t+1}}, \text{AP}_{r_{n-t+2}}, \ldots$.

Consider $\text{AP}_{r_{n-t+1}}$. The accumulated value $\mathcal{A}_{r_{n-t}}$ is contained within the authenticated augmented packet $\text{AP}_{r_{n-t}}$. This value represents the aggregation of list $\mathcal{L}_{r_{n-t}}$ which includes the set $\{P_{r_{n-t}}, P_{r_{n-t+1}}, \ldots, P_{r_{n-t+(n-1)}}\} \cup \{h'(P_1)\}$. Therefore, $\mathcal{A}_{r_{n-t}}$ can be used to authenticate P_1 using MEMBERSHIP since: $r_{(n-t)+1} - r_{n-t} \leq t+1 \leq n-1$.

Once $\mathcal{A}_{r_{(n-t)+1}}$ is authenticated, the receiver can discard $\mathcal{A}_{r_{n-t}}$ and buffer $\mathcal{A}_{r_{(n-t)+1}}$ which will be used to authenticate $\text{AP}_{r_{(n-t)+2}}$ and so on.

This recursive process shows that the receiver can authenticate every packet he obtains, that is to say, the scheme is non-degrading.

B Proof of Theorem 3

Assume that the scheme is either insecure or not (α, β)-correct. By definition, a probabilistic polynomial time opponent $\mathcal{O}$ can break the scheme security or correctness with a non-negligible probability $\pi(k)$ where k is the security parameter setting up the digital signature and the hash function. Note that, since h' is a cryptographic hash function in the RO model, it is assumed to be collision-resistant. We must have either cases:

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

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

In both cases, we will demonstrate that $\mathcal{O}$ can turn an attack against either the correctness or the security of the scheme in polynomial time into forging an element $\widehat{C}$ passing successfully MEMBERSHIP in polynomial time as well.

Point 1. For this attack, $\mathcal{O}$ will have access to the signing algorithm Sign_{SK} (but $\mathcal{O}$ will not have access to SK itself). He can use the public key PK as well as the hash function h'. $\mathcal{O}$ will be allowed to run AUTHENTICATOR whose queries are written as $(\text{BID}_i, n_i, \alpha_i, \beta_i, ¶_i, r_i, \text{DP}_i)$ where DP_i is the set of n_i data packets to be authenticated. As said in Section 4.2, the knowledge of r_i determines the value of k_i as the digest output by h' are $(k_i \log_2(r_i))$-bit long with $\mathcal{H}' = k_i \log_2(r_i)$. In order to get the corresponding output, the signature is obtained by querying Sign_{SK} as a black-box at Step 2 of AUTHENTICATOR.

According to our hypothesis, $\mathcal{O}$ broke the correctness of the authentication protocol. This means that, following the previous process, $\mathcal{O}$ obtained values $\text{BID}, n, \alpha, \beta, \mathcal{P}, r$ and a set of received packets RP such that:

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

It should be noticed that the values $n, \alpha, \beta, \mathcal{P}, r$ as well as PK are publicly known.

Since the digital signature is unforgeable and the hash function is collision resistant, it is impossible to obtain either a forgery (digital signature) or a collision (hash function) in polynomial time with non-negligible probability $\pi(k)/2$. This observation will be used to reduce the security of the authentication scheme to the security of the accumulator.

Since $|\text{RP} \cap \text{AP}| \geq \lceil \alpha\, n \rceil$ and $|\text{RP}| \leq \lfloor \beta\, n \rfloor$, Step 1 of DECODER ends successfully. The consistency of Poly-Reconstruct involves that the list returned by MPR at Step 2 contains the element $\mathcal{A} \| \sigma$ corresponding to DP after removing the pad. It should be noticed that the pad length can be uniquely determined from the public values as $\widetilde{q} = \left\lceil \frac{r+\mathcal{S}+\xi}{\rho\, n+1} \right\rceil$ (see Inequality (1)).

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

$$\widehat{\mathcal{A}} = \mathcal{A}$$

At the beginning of Step 4, the receiver has recovered the accumulated value $\mathcal{A}$ corresponding to the original codeword $(C_1 \cdots C_n)$ related to DP.

Assume that $\mathcal{O}$ cannot forge any $\widehat{C} \notin \{C_1, \ldots, C_n\}$ passing successfully MEMBERSHIP.

In this case, only elements from $\text{RP} \cap \text{AP}$ will successfully pass MEMBERSHIP. As a consequence, at the end of Step 4, we get:

$$\forall i \in \{1, \ldots, n\}\ C'_i \in \{\emptyset, C_i\}$$

and at least $\lceil \alpha\, n \rceil$ values C'_i's are non-empty.

Thus, at Step 5, $(C'_1 \cdots C'_n)$ is first corrected into $(C_1 \cdots C_n)$ and then decoded as $(M_1 \cdots M_{\lceil \alpha\, n \rceil})$. Finally, at the end of Step 6, we have: $\forall i \in \{1, \ldots, n\} \quad P'_i = P_i$. We obtain a contradiction with our original hypothesis which stipulated:

$$\exists j \in \{1, \ldots, n\} \quad P'_j \neq P_j$$

Therefore, $\mathcal{O}$ was able to construct a new value $\widehat{C}$ passing MEMBERSHIP successfully with non-negligible probability in polynomial time.

Point 2. We consider the same kind of reduction as in Point 1. The opponent $\mathcal{O}$ breaks the security of the scheme if one of the following holds:

I. AUTHENTICATOR was never queried on input $\text{BID}, n, \alpha, \beta, \mathcal{P}, r$ and the decoding algorithm DECODER does not reject RP, i.e. $\{P'_1, \ldots, P'_n\} \neq \emptyset$ where:
$\{P'_1, \ldots, P'_n\} = \text{DECODER}(\text{PK}, \text{BID}, n, \alpha, \beta, \mathcal{P}, r, \text{RP})$.

II. AUTHENTICATOR was queried on input $\text{BID}, n, \alpha, \beta, \mathcal{P}, r$ for some data packets $\text{DP} = \{P_1, \ldots, P_n\}$. Nevertheless, the output of DECODER verifies $P'_j \neq P_j$ for some $j \in \{1, \ldots, n\}$.

Case I. Since DECODER output some non-empty packets, Step 3 had to terminate successfully. Thus, it has been found a pair $(h'(\text{BID} \| n \| \alpha \| \beta \| \mathcal{P} \| r \| \mathcal{A}), \sigma)$ such that:

$$\text{Verify}_{\text{PK}}(h'(\text{BID} \| n \| \alpha \| \beta \| \mathcal{P} \| r \| \mathcal{A}), \sigma) = \text{TRUE}$$

If $\mathcal{O}$ never queried AUTHENTICATOR for block tag BID, then either the previous pair is a forgery of the digital signature or $\text{BID} \| n \| \alpha \| \beta \| \mathcal{P} \| r \| \mathcal{A}$ collides with one of the queries $\text{BID}_i \| n_i \| \alpha_i \| \beta_i \| \mathcal{P}_i \| r_i \| \mathcal{A}_i$ made by $\mathcal{O}$ for the hash function h'. Since none of those cases can occur in polynomial time with non-negligible probability, we get a contradiction in this situation.

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

$$(\hat{n}, \hat{\alpha}, \hat{\beta}, \hat{\mathcal{P}}, \hat{r}) \neq (n, \alpha, \beta, \mathcal{P}, r)$$

We conclude as above. That is to say that we get a contradiction with the security of either the digital signature of the hash function.

Case II. We have the same situation as in Point 1.