An Hybrid Approach for Efficient Multicast Stream Authentication over Unsecured Channels

Christophe Tartary[1,2], Huaxiong Wang[1,3], and Josef Pieprzyk[3]

[1] Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University
Singapore
[2] Institute for Theoretical Computer Science
Tsinghua University
Beijing, 100084
P.R. China
[3] Centre for Advanced Computing - Algorithms and Cryptography
Department of Computing
Macquarie University
NSW 2109 Australia
{ctartary,josef}@ics.mq.edu.au,
HXWang@ntu.edu.sg

Abstract. We study the multicast stream authentication problem when an opponent can drop, reorder and inject data packets into the communication channel. In this context, bandwidth limitation and fast authentication are the core concerns. Therefore any authentication scheme is to reduce as much as possible the packet overhead and the time spent at the receiver to check the authenticity of collected elements. Recently, Tartary and Wang developed a provably secure protocol with small packet overhead and a reduced number of signature verifications to be performed at the receiver.

In this paper, we propose an hybrid scheme based on Tartary and Wang's approach and Merkle hash trees. Our construction will exhibit a smaller overhead and a much faster processing at the receiver making it even more suitable for multicast than the earlier approach. As Tartary and Wang's protocol, our construction is provably secure and allows the total recovery of the data stream despite erasures and injections occurred during transmission.

Keywords: Stream Authentication, Polynomial Reconstruction, Unsecured Channel, Merkle Hash Tree, Erasure Code.

1 Introduction

With the expansion of communication networks, broadcasting has become a major technology to distribute digital content from a single user to a large audience via a public communication channel such as the Internet for instance. Online games, military defense systems, satellite television and financial quotes are a few examples of multicast distribution of information. Nevertheless, in large-scale broadcasts, a lost piece of a data stream[1] could generate a flood of retransmission requests from the receivers that

---

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

congregate at the sender's side. Furthermore the network can be under the influence of malicious users performing illegal and damaging operations on the stream. As a consequence, the security of a multicast authentication protocol relies on the network properties and the opponents' computational power. Several unconditionally secure schemes have been developed [5, 9, 36] but either these are one-time protocols or they require too large storage capacities. In this work, we consider that adversaries have polynomially bounded computational abilities.

An application like a pay-TV channel broadcasting programs 24 hours a day and seven a week suggests that the stream can be considered as infinite. Nevertheless the receivers must be able to authenticate data within a short period of time upon reception. Since many protocols will distribute private or sensitive content, non-repudiation of the sender is required for most of them as using data from an uncertain origin can have disastrous consequences during military operations for instance. Unfortunately signing each data packet[2] is impractical as digital signatures are generally very expensive to generate and/or verify. Furthermore bandwidth limitations prevent one-time and k-time signatures [11, 35] from being used due to their large size. Boneh et al. constructed short signatures in [6] but their verification time is prohibitive to be a practical solution for authenticated broadcast [3, 37]. Thus a general approach is to generate a single signature and to amortize its computational cost and overhead over several data packets using a chain of hash functions for instance.

Several constructions relying on hash functions have been developed to deal with packet loss [12, 21, 31, 32]. A signature is generated from time to time and is always assumed to be received correctly. This provides authentication and non-repudiation of the sender and allows new receivers to join the communication group at any block[3] boundary. Using Markov chains [10, 30, 42] to model the network packet loss, the authors of the previous constructions determined bounds on the packet authentication probability. Unfortunately, the main issue in those schemes is the fact that they rely on the reliable reception of signature packets. Since networks like the Internet only provide a best effort delivery of data, the reliability requirement limits the area of applications of those constructions.

In order to overcome this issue, a general solution is to split the signature into k parts where only ℓ of them ($\ell < k$) are enough to guarantee the recovery of the whole signature. Many schemes have been developed using this idea [1, 26, 27, 28, 29] but none of them tolerates a single packet injection. Using a Merkle hash tree [20], Wong and Lam developed a construction dealing with both erasures and injections [41]. Nevertheless, it is vulnerable to denial of service attacks (DoS) against the computational resources of the receiver as each packet carries the block signature. Thus, in the worst case, the number of signature verifications to be performed per block of n packets is $\Theta(n)$. In [15], Karlof et al. overcame this problem by using Merkle hash trees as one-way accumulators [2, 4, 24, 25]. Their approach requires $O(1)$ signature verifications per block in any case and each augmented packet[4] has to carry $\lceil \log_2 n \rceil$ hashes which may be too

---

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

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

[4] 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.

large for resource limited receivers. In [17], Lysyanskaya et al. used a polynomial representation as well as an algorithm by Guruwami and Sudan [14] to deal with packet drops and data injections. As in Karlof et al.'s construction, their technique requires $O(1)$ signature verifications but its packet overhead is $O(1)$ bits. Recently, Lysyankaya et al.'s approach has been extended by Tartary and Wang [39]. This scheme uses Maximum Distance Separable (MDS) codes [18] and is denoted as TWMDS in our paper. It requires $O(1)$ signature verifications per block, $O(1)$ bit packet overhead and enables all data packets to be recovered at the receiver despite erasures and injections thanks to the erasure correcting code. This feature is important when the application processing the data packets is not loss tolerant as it may be the case for military applications where obtaining all information about the enemy target is vital or for high quality video streaming where this technique prevents frozen images to happen.

In this work, we present an hybrid construction based on Merkle hash trees and TWMDS which will be provably secure in the random oracle model. This idea of using a Merkle hash tree for multicast authentication is note new but this technique enables fast authentication as only hash computations are performed. As in TWMDS, our new scheme will enable the whole data packets to be recovered at the receiver despite erasures and injections and will allow new members to join the communication group at any block boundary. As noted earlier, both the packet overhead and the speed of authentication at the receiver are the core concerns for multicast stream authentication. Since the relation between overhead and speed is central in this context and limits the scope of applications of many schemes, we will emphasize that our protocol exhibits a smaller overhead and a much faster authentication process than TWMDS making our scheme more suitable for broadcast applications. As TWMDS, the non-repudiation of the stream origin will be guaranteed using a digital signature.

The plan of this paper is as follows. In the next section, we introduce our network model and recall a few results from [14]. Our authentication scheme is described in Sect. 3 while its security and recovery property will be discussed in Sect. 4. In Sect. 5, we present the benefits of our approach in term of overhead as well as authentication speed at the receiver. Finally, we will sum up our contribution to the multicast authentication problem over unsecured channels in Sect. 6.

2   Preliminaries

We now present our network model as well as an erasure correcting code we use in our construction. We also recall a modified version of the algorithm Poly-Reconstruct by Guruswami and Sudan [14] which will be used to deal with data injections and packet drops as in [17, 39].

Network Model. We consider that the communication channel is under the influence of an opponent $\mathcal{O}$ who can drop and rearrange packets of his choice as well as can inject bogus data into the network. This corresponds to the unsecured communication channel model described by Menezes et al. in [19]. We investigate the multicast stream authentication problem. Thus we can assume that a reasonable number of original augmented packets reaches the receivers and not too many incorrect chunks of data are injected by $\mathcal{O}$. Indeed, if too many original packets are dropped then data transmission becomes

the main problem to treat since a small number of received packets would be probably useless even if authenticated. On the other hand, if $\mathcal{O}$ injects a large number of forged packets then the main problem becomes increasing the resistance against DoS attacks. In order to build our signature amortization scheme, we need to split the data stream into blocks of n packets: $P_1, \ldots, P_n$. 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.

Code Construction. In our construction, we focus on linear codes to correct erasures. As in [39], we use *Maximum Distance Separable* (MDS) codes [18]. As our scheme works with any MDS code, we refer the reader to [39] for a discussion about which family of MDS codes to choose for best efficiency. Note that any linear code can be represented by a *generator matrix* G. *Encoding* a message m (represented as a row vector) means computing the corresponding codeword c as: $c := m\,G$ (see [18]).

Polynomial Reconstruction Algorithm. In [14], Guruswami and Sudan developed an algorithm Poly-Reconstruct to solve the polynomial reconstruction problem. They proved that if T points were given as input then their algorithm 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 the same version of Poly-Reconstruct as in [39] where it was named MPR. Denote $\mathbb{F}_{2^q}$ the field representing the coefficients of the polynomial. Every element of $\mathbb{F}_{2^q}$ can be represented as a polynomial of degree at most $q - 1$ over $\mathbb{F}_2$ (see [16]). Operations in $\mathbb{F}_{2^q}$ are performed modulo a polynomial $\mathcal{Q}(X)$ of degree q ($\mathcal{Q}(X)$ is irreducible over $\mathbb{F}_2$).

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

Note that Poly-Reconstruct runs in time quadratic in N and outputs a list of size at most quadratic in N as well (see Theorem 6.12 and Lemma 6.13 from [13]). Algorithms for implementing Poly-Reconstruct can be found in [22].

3 Our Hybrid Authentication Protocol

In order to guarantee the security of our construction, we need a collision resistant hash function h (see [33]) and an unforgeable signature scheme $(\text{Sign}_{\text{SK}}, \text{Verify}_{\text{PK}})$ (see [38]) the key pair of which (SK,PK) is created by a generator KeyGen as in [15, 17, 39].

Scheme Overview. Each block contains n data packets $P_1, \ldots, P_n$ and is located within the whole stream using its identification value BID. Our algorithms apply two steps.

The first step works as follows. Due to our network model, we want to generate n augmented packets $AP_1, \ldots, AP_n$ such that we can reconstruct the sequence of packets $P_1, \ldots, P_n$ from any $\lceil \alpha n \rceil$-subset of $\{AP_1, \ldots, AP_n\}$. Thus we need to encode $P_1, \ldots, P_n$ using a code which can correct up to $n - \lceil \alpha n \rceil$ erasures. Therefore we employ a $[n, \lceil \alpha n \rceil, n - \lceil \alpha n \rceil + 1]$ code. Notice that the use of such a code implies that the elements of the code alphabet are larger than the size of a data packet as the message to be encoded $(M_1 \cdots M_{\lceil \alpha n \rceil})$ should represent the concatenation $P_1 \| \cdots \| P_n$.

The second step of our algorithm consists of building Merkle hash trees. If we denote $(C_1 \cdots C_n)$ the codeword corresponding to the message $(M_1 \cdots M_{\lceil \alpha n \rceil})$ then we partition the digests $h(C_1), \ldots, h(C_n)$ into f families of $\lceil \frac{n}{f} \rceil$ elements where f is an efficiency parameter (see Sect. 5). Remark that if f does not divide n then the last family will be completed with dummy packets (consisting of zeros for simplicity). This family padding has no effect on the number of augmented packets sent into the network as those dummy elements will only be used to construct the last family tree. Since f and n will be public, each receiver knows how many dummy packets to add for the last family. For each family $F_j := \{h(C_{(j-1)\lceil \frac{n}{f} \rceil + 1}), \ldots, h(C_{j \lceil \frac{n}{f} \rceil})\}$ (for $j \in \{1, \ldots, f\}$) we build the Merkle hash tree the leaves of which are the elements of F_j (see Fig. 1 for an example).
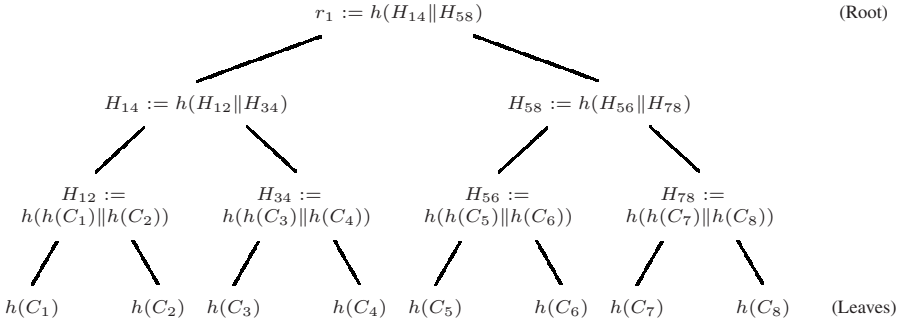
Fig. 1. the Merkle hash tree of F_1 when $\lceil \frac{n}{f} \rceil = 8$

To provide authentication and non-repudiation and allow new members to join the communication group at block boundaries, we sign the digest $h(r_1 \| \cdots \| r_f)$ where $r_1, \ldots, r_f$ are the f tree roots. As in [39], we construct a polynomial $A(X)$ of degree at most ρn (for some rational constant ρ), the coefficients of which represent $r_1 \| \cdots \| r_f \| \sigma$ where σ is the signature. We build the augmented packets as:

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

where $\text{path}(i)$ denotes the $\lceil \log_2 \lceil \frac{n}{f} \rceil \rceil$ hashes needed to reconstruct the path from $h(C_i)$ to the root of his family tree. For instance on Fig. 1, we have $\text{path}(2) = h(C_1) \| H_{34} \| H_{58}$. As said earlier, BID denotes the position of the block $P_1, \ldots, P_n$ within the stream.

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 tree roots $r_1, \ldots, r_f$. Thus he can identify the correct C_i's amongst the list of elements he got by checking which paths are correct within the f trees. 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 $P_1, \ldots, P_n$.

Formal Scheme Construction. As in [39], we assume that α and β are rational numbers so that we can represent them over a finite number of bits using their numerator and denominator. In order to run Poly-Reconstruct as a part of MPR, we have to choose $\rho \in (0, \frac{\alpha^2}{\beta})$. Remark that it is suggested in [39] to choose $\rho = \frac{\alpha^2}{2\beta}$ to get a small list returned by Poly-Reconstruct. Notice that ρ has to be rational since ρn is an integer. We also consider that the $[n, \lceil \alpha n \rceil, n - \lceil \alpha n \rceil + 1]$ code is uniquely determined (i.e. its generator matrix G is known) when n, α, β and ρ are known. Denote $\mathbb{F}_{2^{\tilde{q}}}$ the field of this MDS code. Due to space limitations, we omitted the construction of $q, \tilde{q}$ as well as the different pads used by our scheme which can be found in the full version of this paper. Table 1 summarizes the scheme parameters which are assumed to be publicly known.

Table 1. Public parameters for our authentication scheme

n: Block length	$\mathcal{Q}(X)$: Polynomial representing the field for the MDS code
f: Number of families	$\mathcal{P}$: bit size of data packets
α, β: Network rates	G: Generating matrix of the MDS code
ρ: Ratio	$\mathcal{Q}(X)$: Polynomial representing the field for polynomial interpolation

The hash function h as well as the signature verification algorithm Verify and the signature public key 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 [23] while the digital signature is a 1024-bit RSA signature [34]. We denote $\mathcal{H}$ the digest bit length and $\mathcal{S}$ the bit length of a signature. Since h and the digital signature are publicly known, so are $\mathcal{H}$ and $\mathcal{S}$.

Authenticator
Input: The secret key SK, the block number BID, Table 1 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.

/* Tree Construction */

2. For j from 1 to f do

Compute the digests $h(C_{(j-1)\lceil \frac{n}{f} \rceil + 1}), \ldots, h(C_{j \lceil \frac{n}{f} \rceil})$ and build the Merkle hash tree having the previous digests as leaves (as said earlier some padding with zeros values may be needed when $j = f$). Denote r_j its root.

/* Signature Generation */

3. Write R as $R := r_1\|\cdots\|r_f$. Compute the family signature σ as $\sigma := \mathrm{Sign}_{\mathrm{SK}}(h(\mathrm{BID}\|f\|n\|\alpha\|\beta\|\mathcal{P}\|R))$. Parse $R\|\sigma$ as $a_0\|\cdots\|a_{\rho n}$ where each $a_i \in \mathbb{F}_{2^q}$ after padding.

4. Construct the block polynomial $A(X) := a_0 + a_1 X + \cdots + a_{\rho n} X^{\rho n}$ and evaluate it at the first n points[5] of $\mathbb{F}_{2^q}$.

/* Construction of Augmented Packets */

6. Build the augmented packet AP_i as $\mathrm{AP}_i := \mathrm{BID}\|i\|C_i\|A(i)\|$ path(i) where path(i) is defined as in the scheme overview section.

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

As in [39], assuming that α and β are rational enabled us to write $\alpha\|\beta$ over a finite number of bits. It should be noticed that when $(n, f, \alpha, \beta, \mathcal{P}, \rho)$ are given, each step of Authenticator is uniquely determined as soon as $(\mathcal{Q}(X), G, \widetilde{\mathcal{Q}}(X))$ are provided. Furthermore since 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. For instance, consider the remark made in [39] where ρ is suggested to be set as $\frac{\alpha^2}{2\beta}$. As a consequence, we can consider that when $(n, f, \alpha, \beta, \mathcal{P})$ are given, $(\rho, \mathcal{Q}(X), G, \widetilde{\mathcal{Q}}(X))$ are uniquely determined. This consideration is identical to what is assumed in [39].

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

/* Signature Verification and Root Recovery */

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

2. Rename the remaining elements as $\{\mathrm{AP}'_1, \ldots, \mathrm{AP}'_{\mathcal{N}}\}$ and write each element as: $\mathrm{AP}'_i = \mathrm{BID}\|j_i\|C'_{j_i}\|A_{j_i}\|$ path$'_{j_i}$ where $j_i \in \{1, \ldots, n\}$. Run MPR on the set $\{(j_i, A_{j_i}), 1 \leq i \leq \mathcal{N}\}$ to get a list $L := \{\mathcal{C}_1, \ldots, \mathcal{C}_\mu\}$ of candidates for signature verification. If MPR rejects that set then the algorithm stops.

3. Set $r'_k = \emptyset$ for $k \in \{1, \ldots, f\}$. While the signature has not been verified and the list L has not been exhausted, pick a new candidate $\tilde{r}_1\|\cdots\|\tilde{r}_f\|\tilde{\sigma}$ after removing the pad. If $\mathrm{Verify}_{\mathrm{PK}}(h(\mathrm{BID}\|f\|n\|\alpha\|\beta\|\mathcal{P}\|\tilde{r}_1\|\cdots\|\tilde{r}_f), \tilde{\sigma}) = \mathrm{TRUE}$ then $\tilde{\sigma}$ is considered as the authentic block signature σ and we set $r'_k = \tilde{r}_k$ for $k \in \{1, \ldots, f\}$ as authentic tree roots. If L is exhausted before the signature is verified then our algorithm stops.

/* Packet Decoding */

4. Set $\mathcal{C}' := (\emptyset, \ldots, \emptyset)$. For each of the $\mathcal{N}$ remaining packets, $\mathrm{BID}\|j_i\|C'_{j_i}\|A_{j_i}\|$ path$'_{j_i}$,

---

[5] Any element of $\mathbb{F}_{2^q}$ can be represented as $\lambda_0 Y^0 + \lambda_1 Y_1 + \ldots + \lambda_{q-1} Y^{q-1}$ where each λ_i belongs to $\mathbb{F}_2$. We 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 $n - 1$.

we first compute its family number ℓ_{j_i} as $\ell_{j_i} := \left\lceil \frac{j_i}{\lceil n/f \rceil} \right\rceil$. Second, if the path from $h(C'_{j_i})$ to the value $r'_{\ell_{j_i}}$ can be reconstructed using path'_{j_i} then we set the j_i^{th} coordinate of $\mathcal{C}'$ to C_{j_i}.

5. If $\mathcal{C}'$ has less than $\lceil \alpha\,n \rceil$ non-erased coordinates then the algorithm stops.
   Else
   5.1. Correct the erasures of $\mathcal{C}'$ using the MDS decoding process and denote $(M'_1, \ldots, M'_{\lceil \alpha\,n \rceil})$ the corresponding message.
   5.2. Remove the pad from $M'_1 \| \cdots \| M'_{\lceil \alpha\,n \rceil}$ and write the resulting string as $P'_1 \| \cdots \| P'_n$.

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

Note that when Decoder stops then the whole content of block BID is lost. Nevertheless the definitions of α and β ensure that this will never happen (see Theorem 2).

4   Security and Recovery Analysis

Security of the Scheme. We recall the security definition as presented in [39].

Definition 1 ([39]). (KeyGen,Authenticator,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* $(\text{BID}, f, n, \alpha, \beta, \mathcal{P}, \rho, \mathcal{Q}(X), \widetilde{\mathcal{Q}}(X), G, \text{RP})$.

$\mathcal{O}$ *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 to identify all the correct packets.*
  b. *(security violation)* $\mathcal{O}$ *succeeds to output* RP *such that Decoder outputs* $\{P'_1, \ldots, P'_n\}$ *that was never authenticated by* Authenticator *for parameters* $(\text{BID}, f, n, \alpha, \beta, \mathcal{P}, \rho, \mathcal{Q}(X), \widetilde{\mathcal{Q}}(X), G)$.

We now show that our construction also satisfies the above security definition. The proof of the following theorem can be found in Appendix A.

Theorem 1. *Our scheme* (KeyGen,Authenticator,Decoder) *is secure and* (α, β)*-correct.*

Recovery Property. We now show that our scheme enables any receiver to recover the n data packets and the number of signature verifications to be performed per block is upper bounded by the same value as for TWMDS. We recall the following definition:

Definition 2 ([39]). *Given a flow of n symbols, we say that the survival and flood rates (α, β) are accurate if: (1) data are sent per block of n elements through the network and (2) for any block of n elements $\{E_1, \cdots, E_n\}$ emitted by the sender, if we denote $\{\tilde{E}_1, \ldots, \tilde{E}_\mu\}$ the set of received packets then $\mu \leq \lfloor \beta n \rfloor$ and at least $\lceil \alpha n \rceil$ elements of $\{E_1, \cdots, E_n\}$ belong to $\{\tilde{E}_1, \ldots, \tilde{E}_\mu\}$. Condition (2) must be true for each receiver belonging to the communication group.*

From this point onwards, we assume that (α, β) is accurate for our network flow n. As in [39], we have the following result whose proof can be found in Appendix B.

Theorem 2. *For any BID, each receiver recovers the n original data packets $P_1, \ldots, P_n$. 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:*

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

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

5   Efficiency Analysis

As said in Sect. 1, bandwidth limitations and authentication delay are two major concerns for authentication protocols. In this section we will see that for a suitable choice of f our construction can achieve smaller overhead than TWMDS while exhibiting a much faster authentication at the receiver.

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

Our augmented packets are written as $\text{BID}\|i\|C_i\|A(i)\|\,\text{path}(i)$. The element C_i is represented by $\left\lceil \frac{n\mathcal{P}}{\lceil \alpha n \rceil} \right\rceil$ bits while $A(i)$ requires $\left\lceil \frac{f\mathcal{H}+\mathcal{S}+\lambda}{\rho n+1} \right\rceil$ bits where λ is the smallest element of $\mathbb{N}$ such that:

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

is verified (see the full version of the paper for details). The element $\text{path}(i)$ consists of $\left\lceil \log_2 \lceil \frac{n}{f} \rceil \right\rceil$ digests computed by h. Therefore, our packet overhead ω is equal to:

$$\omega := \left\lceil \frac{n\mathcal{P}}{\lceil \alpha n \rceil} \right\rceil - \mathcal{P} + \left\lceil \frac{f\mathcal{H}+\mathcal{S}+\lambda}{\rho n+1} \right\rceil + \left\lceil \log_2 \lceil \frac{n}{f} \rceil \right\rceil \mathcal{H} \text{ bits}$$

The augmented packets of TWMDS are written as $\text{BID}\|i\|C_i\|A(i)$ where C_i is represented over $\left\lceil \frac{n\mathcal{P}}{\lceil \alpha n \rceil} \right\rceil$ bits while $A(i)$ requires $\left\lceil \frac{n\mathcal{H}+\mathcal{S}}{\rho n+1} \right\rceil$ bits. Therefore, the overhead ω_{TWMDS} of TWMDS is equal to:

$$\omega_{\text{TWMDS}} := \left\lceil \frac{n\,\mathcal{P}}{\lceil \alpha\,n \rceil} \right\rceil - \mathcal{P} + \left\lceil \frac{n\,\mathcal{H} + \mathcal{S}}{\rho\,n + 1} \right\rceil \text{ bits}$$

To illustrate the benefits of our approach over TWMDS, we will compute the ratio $\frac{\omega}{\omega_{\text{TWMDS}}}$ for different choices of $\mathcal{P}, \alpha, \beta$. We choose the network rates as in [39] and the packet size to 512 and 4096 bits as in [32]. We pick $n = 1000$ as in those two works and set ρ to $\frac{\alpha^2}{2\beta}$ as suggested in [39]. We used SHA-256 as a hash function and a 1024-bit RSA signature scheme. The first step is to compute f minimizing our overhead ω. These values are shown in Table 2 while the corresponding overhead ω is in Table 3.

Table 2. Number of families f minimizing ω when $n = 1000$

| | | $\mathcal{P} = 512$ | | | | $\mathcal{P} = 4096$ | | | |
| | | α | | | | α | | | |
		0.5	0.75	0.8	0.9	0.5	0.75	0.8	0.9
β	1.1	125	500	500	500	125	500	500	500
	1.25	125	250	500	500	125	250	500	500
	1.5	125	250	250	500	125	250	250	500
	2	125	250	250	250	125	250	250	250

Table 3. Overhead of our construction ω when f is chosen as in Table 2 and $n = 1000$

| | | $\mathcal{P} = 512$ | | | | $\mathcal{P} = 4096$ | | | |
| | | α | | | | α | | | |
		0.5	0.75	0.8	0.9	0.5	0.75	0.8	0.9
β	1.1	1569	930	827	663	5153	2125	1723	1062
	1.25	1607	971	887	710	5191	2166	1783	1109
	1.5	1672	1028	944	790	5256	2223	1840	1189
	2	1801	1143	1044	889	5385	2338	1940	1288

Our comparison to TWMDS is depicted in Table 4. It clearly shows that our construction exhibits a smaller overhead than TWMDS. Our benefits get larger over networks with small reliability (i.e. α is small) or highly polluted by $\mathcal{O}$ (i.e. β is large). Our construction also seems to perform even better when the data packets are small.

Table 4. Ratio $\frac{\omega}{\omega_{\text{TWMDS}}}$ when $n = 1000$

| | | $\mathcal{P} = 512$ | | | | $\mathcal{P} = 4096$ | | | |
| | | α | | | | α | | | |
		0.5	0.75	0.8	0.9	0.5	0.75	0.8	0.9
β	1.1	56.95%	79.28%	81.96%	87.93%	81.29%	89.74%	90.45%	92.11%
	1.25	52.57%	74.18%	78.57%	83.73%	78.17%	86.50%	88.05%	88.93%
	1.5	46.97%	66.97%	71.08%	78.53%	73.57%	81.43%	82.73%	84.63%
	2	39.50%	57.55%	60.52%	67.30%	66.12%	73.50%	74.02%	74.88%

Authentication Efficiency. We now compare the authentication delay at the receiver between our construction and TWMDS. It should be noticed that the authentication part of Decoder consists of Steps 1 to 4. Indeed Step 5 is dedicated to erasure correction which is a non-authentication related feature. In those four steps, two points matter: the number of signature verification queries and the quantity of information to be processed by h. In our comparison, we focus on the worst case, i.e. we assume that the number of signature verification is $U(n)$. Note this is the same value as in [39]. In this situation, the number of bits $\mathfrak{h}_1$ processed by h is:

$$U(n)\left(|\mathrm{BID}|+f\,\mathcal{H}+|f|+\lceil\log_2 n\rceil+|\alpha|+|\beta|+\lceil\log_2\mathcal{P}\rceil\right)+\lfloor\beta\,n\rfloor\left(\mathcal{P}+2\left\lceil\log_2\lceil\tfrac{n}{f}\rceil\right\rceil\mathcal{H}\right)$$

As $f \leq n$, we can assume that $|f| = \lceil\log_2 n\rceil$ bits. Considering TWMDS, the number of bits $\mathfrak{h}_2$ processed by the hash function is:

$$U(n)\left(|\mathrm{BID}| + n\,\mathcal{P} + \lceil\log_2 n\rceil + |\alpha| + |\beta| + \lceil\log_2\mathcal{P}\rceil\right) + \lfloor\beta\,n\rfloor\,\mathcal{P}$$

Table 5 represents the ratio $\mathcal{T} := \dfrac{U(n)\,t_{\mathcal{S}} + \mathfrak{h}_1\,t_{\mathcal{H}}}{U(n)\,t_{\mathcal{S}} + \mathfrak{h}_2\,t_{\mathcal{H}}}$ where $t_{\mathcal{S}}$ denotes the number of seconds required to perform one signature verification and $t_{\mathcal{H}}$ is the number of seconds to hash one bit. In [31], it is assumed that there are about 500 packets sent per second in the network in the case of video broadcast. As $n = 1000$, we buffer roughly 2 seconds of video per block. So if BID is represented over 30 bits, then it provides a stream which can last at least 68 years. It is also realistic to assume that $|\alpha|$ and $|\beta|$ are negligible in comparison to $|\mathrm{BID}|$, $\mathcal{H}$, $\lceil\log_2 n\rceil$ and $\lceil\log_2\mathcal{P}\rceil$. Our results are based on Dai's benchmarks [8].

Table 5. Ratio $\mathcal{T}$ when $n = 1000$

		$\mathcal{P} = 512$				$\mathcal{P} = 4096$			
		α				α			
		0.5	0.75	0.8	0.9	0.5	0.75	0.8	0.9
	1.1	37.98%	54.08%	56.98%	60.85%	10.49%	19.14%	21.18%	23.89%
β	1.25	38.31%	56.40%	56.46%	59.64%	10.62%	19.41%	20.82%	23.04%
	1.5	38.14%	55.22%	58.35%	60.26%	10.55%	18.84%	20.34%	23.48%
	2	37.93%	53.85%	58.35%	68.25%	10.47%	18.19%	20.34%	25.03%

Table 5 shows that our construction is much faster than TWMDS. Note that we deliberately removed the query to Poly-Reconstruct happening at Step 2 of Decoder. Nevertheless TWMDS also performs such a request. So if the time needed to run Poly-Reconstruct is added to both numerator and denominator of $\mathcal{T}$ then the values of Table 5 will be flattened but our scheme will nonetheless remain faster.

In [40], Tartary and Wang suggested to use the provably collision resistant trapdoor hash function Very Smooth Hash (VSH) [7] instead of a digital signature to speed up the running time at the receiver. Based on Contini et al.'s work, VSH is 25 times slower than SHA-1 while it requires to use a 1516-bit modulus to achieve the same security level as a 1024-bit RSA signature modulus. Table 6 describes the overhead for our construction,

Table 6. Minimal overhead of our construction for $n = 1000$ when using VSH

		$\mathcal{P} = 512$				$\mathcal{P} = 4096$			
		α				α			
		0.5	0.75	0.8	0.9	0.5	0.75	0.8	0.9
β	1.1	1573	932	828	664	5157	2127	1724	1063
	1.25	1612	973	888	712	5196	2168	1784	1111
	1.5	1678	1031	946	791	5262	2226	1842	1190
	2	1808	1146	1047	891	5392	2341	1943	1290

Table 7. Ratio $\dfrac{\omega}{\omega_{\text{TWMDS}}}$ for $n = 1000$ when using VSH

		$\mathcal{P} = 512$				$\mathcal{P} = 4096$			
		α				α			
		0.5	0.75	0.8	0.9	0.5	0.75	0.8	0.9
β	1.1	57.10%	79.45%	82.06%	88.06%	81.35%	89.82%	90.50%	92.19%
	1.25	52.73%	74.33%	78.65%	83.96%	78.24%	86.58%	88.10%	89.09%
	1.5	47.13%	67.17%	71.23%	78.63%	73.66%	81.54%	82.82%	84.70%
	2	39.65%	57.70%	60.70%	67.45%	66.21%	73.59%	74.13%	75.00%

Table 8. Ratio $\mathcal{T}$ for $n = 1000$ when using VSH

		$\mathcal{P} = 512$				$\mathcal{P} = 4096$			
		α				α			
		0.5	0.75	0.8	0.9	0.5	0.75	0.8	0.9
β	1.1	26.81%	43.71%	46.82%	50.97%	8.95%	17.72%	19.79%	22.54%
	1.25	27.15%	46.02%	46.26%	49.74%	9.07%	17.98%	19.42%	21.68%
	1.5	26.97%	44.79%	48.06%	50.33%	9.01%	17.41%	18.92%	22.12%
	2	26.75%	43.36%	48.06%	58.38%	8.92%	16.74%	18.92%	23.68%

Table 7 depicts the ratio $\dfrac{\omega}{\omega_{\text{TWMDS}}}$ and Table 8 represents the speed ratio $\mathcal{T}$ when VSH is used instead of RSA.

One notices that using VSH slightly increases the overhead with respect to the digital signature approach but it reduces the authentication time at the receiver even further.

6   Conclusion

In this paper, we presented an hybrid construction based on Merkle hash trees and TWMDS. Our scheme is provably secure under the random oracle model and enables new participants to join the communication group at every block boundary. As TWMDS, our approach allows the whole data packets to be recovered at the receiver. The tradeoff between overhead and authentication speed limits the application of many constructions. The main benefits of this interaction between MDS codes and Merkle hash tree is that our packet overhead and authentication speed are much smaller than for TWMDS. If the number of families f is suitably chosen then, when using 512-bit packets, our overhead

is between 39% and 88% of TWMDS while the authentication speed is between 66% and 92%. When using larger packets, the benefits of our construction increase even further as the overhead then represents between 38% and 68% of TWMDS while the authentication speed is between 11% and 25%. The advantages of our scheme are important when the reliability of the network is small and the pollution due to the attacker is large.

We also saw that when we employed a trapdoor hash function such a VSH instead of a digital signature as suggested in [40], the benefits of our scheme increased even further.

Acknowledgement

References

[1] Al-Ibrahim, M., Pieprzyk, J.: Authenticating multicast streams in lossy channels using threshold techniques. In: Lorenz, P. (ed.) ICN 2001. LNCS, vol. 2094, pp. 239–249. Springer, Heidelberg (2001)

[2] Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology - Eurocrypt'97*. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997)

[3] Barreto, P.S., Kim, H.Y., Lynn, B., Scott, M.: Efficient algorithms for pairing-based cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–369. Springer, Heidelberg (2002)

[4] Benaloh, J., de Mare, M.: One-way accumulators: A decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)

[5] Blundo, C., De Santis, A., Herzberg, A., Kutten, S., Vaccaro, U., Yung, M.: Perfectly-secure key distribution for dynamic conferences. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 471–486. Springer, Heidelberg (1993)

[6] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)

[7] Contini, S., Lenstra, A.K., Steinfeld, R.: VSH: an efficient and provable collision resistant hash function. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 165–182. Springer, Heidelberg (2006)

[8] Dai, W.: Crypto++ 5.2.1 benchmarks (July 2004)

[9] Desmedt, Y., Frankel, Y., Yung, M.: Multi-receiver/multi-sender network security: Efficient authenticated multicast/feedback. In: IEEE INFOCOM 1992, vol. 3, pp. 2045–2054. IEEE Computer Society Press, Los Alamitos (1992)

[10] Fu, J.C., Lou, W.Y.W.: Distribution Theory of Runs and Patterns and its Applications. World Scientific Publishing, Singapore (2003)

[11] Gennaro, R., Rohatgi, P.: How to sign digital streams. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 180–197. Springer, Heidelberg (1997)

[12] Golle, P., Modadugu, N.: Authenticating streamed data in the presence of random packet loss. In: Symposium on Network and Distributed Systems Security, pp. 13–22. Internet Society (2001)

[13] Guruswami, V.: List Decoding of Error-Correcting Codes. Springer, Heidelberg (2004)

[14] Guruswami, V., Sudan, M.: Improved decoding of Reed-Solomon and algebraic-geometric codes. IEEE Transactions on Information Theory 45(6), 1757–1767 (1999)

[15] Karlof, C., Sastry, N., Li, Y., Perrig, A., Tygar, J.D.: Distillation codes and applications to DoS resistant multicast authentication. In: 11th Network and Distributed Systems Security Symposium (NDSS) (2004)

[16] Lidl, R., Niederreiter, H.: Introduction to Finite Fields and their Applications - Revised Edition. Cambridge University Press, Cambridge (2000)

[17] Lysyanskaya, A., Tamassia, R., Triandopoulos, N.: Multicast authentication in fully adversarial networks. In: IEEE Symposium on Security and Privacy, pp. 241–253. IEEE Computer Society Press, Los Alamitos (2003)

[18] MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error-Correcting Codes. North-Holland, Amsterdam (1977)

[19] Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton, USA (1996)

[20] Merkle, R.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)

[21] Miner, S., Staddon, J.: Graph-based authentication of digital streams. In: IEEE Symposium on Security and Privacy, pp. 232–246. IEEE Computer Society Press, Los Alamitos (2001)

[22] Moon, T.K.: Error Correction Coding: Mathematical Methods and Algorithms. Wiley, Chichester (2005)

[23] National Institute of Standards and Technology. FIPS 180-2: Secure Hash Standard (SHS) (August 2002. Amended 25 (February 2004)), Available online at:
http://csrc.nist.gov/publications/fips/fips180-2/
fips180-2withchangenotice.pdf

[24] Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)

[25] Nyberg, K.: Fast accumulated hashing. In: Gollmann, D. (ed.) Fast Software Encryption. LNCS, vol. 1039, pp. 83–87. Springer, Heidelberg (1996)

[26] Pannetrat, A., Molva, R.: Authenticating real time packet streams and multicasts. In: 7th International Symposium on Computers and Communications, IEEE Computer Society Press, Los Alamitos (2002)

[27] Park, J.M., Chong, E.K.P., Siegel, H.J.: Efficient multicast packet authentication using signature amortization. In: IEEE Symposium on Security and Privacy, pp. 227–240. IEEE Computer Society Press, Los Alamitos (2002)

[28] Park, J.M., Chong, E.K.P., Siegel, H.J.: Efficient multicast stream authentication using erasure codes. ACM Transactions on Information and System Security 6(2), 258–285 (2003)

[29] Park, Y., Cho, Y.: The eSAIDA stream authentication scheme. In: Laganà, A., Gavrilova, M., Kumar, V., Mun, Y., Tan, C.J.K., Gervasi, O. (eds.) ICCSA 2004. LNCS, vol. 3046, pp. 799–807. Springer, Heidelberg (2004)

[30] Paxson, V.: End-to-end Internet packet dynamics. IEEE/ACM Transactions on Networking 7(3), 277–292 (1999)

[31] Perrig, A., Canetti, R., Tygar, J.D., Song, D.: Efficient authentication and signing of multicast streams over lossy channels. In: IEEE Symposium on Security and Privacy, pp. 56–73. IEEE Computer Society Press, Los Alamitos (2000)

[32] Perrig, A., Tygar, J.D.: Secure Broadcast Communication in Wired and Wireless Networks. Kluwer Academic Publishers, Dordrecht (2003)

[33] Pieprzyk, J., Hardjono, T., Seberry, J.: Fundamentals of Computer Security. Springer, Heidelberg (2003)

[34] Rivest, R.L., Shamir, A., Adelman, L.: A method for obtaining digital signatures and public key cryptosystems. Communication of the ACM 21(2), 120–126 (1978)

[35] Rohatgi, P.: A compact and fast hybrid signature scheme for multicast packet authentication. In: 6th ACM Conference on Computer and Communications Security, pp. 93–100. ACM Press, New York (1999)

[36] Safavi-Naini, R., Wang, H.: New results on multi-receiver authentication code. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 527–541. Springer, Heidelberg (1998)

[37] Scott, M., Costigan, N., Abdulwahab, W.: Implementing cryptographic pairings on smartcards. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 134–147. Springer, Heidelberg (2006)

[38] Stinson, D.R.: Cryptography: Theory and Practice, 3rd edn. Chapman & Hall/CRC (2006)

[39] Tartary, C., Wang, H.: Achieving multicast stream authentication using MDS codes. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 108–125. Springer, Heidelberg (2006)

[40] Tartary, C., Wang, H.: Efficient multicast stream authentication for the fully adversarial network. International Journal of Security and Network (Special Issue on Cryptography in Networks) 2(3/4), 175–191 (2007)

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

[42] Yajnik, M., Moon, S., Kurose, J., Towsley, D.: Measurement and modeling of the temporal dependence in packet loss. In: IEEE INFOCOM 1999, vol. 1, pp. 345–352. IEEE Computer Society Press, Los Alamitos (1999)

A   Proof of Theorem 1

Assume that the scheme is either insecure or not (α, β)-correct. By definition an 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. Therefore we must have either cases:

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

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

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

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

For this attack, $\mathcal{O}$ will have access to the signing algorithm Sign_{SK} (but $\mathcal{O}$ will not have access to SK itself). He can use the public key PK as well as the collision resistant hash function h. $\mathcal{O}$ will be allowed to run Authenticator whose queries are written as $(\text{BID}_i, f_i, n_i, \alpha_i, \beta_i, \mathcal{P}_i, \rho_i, \mathcal{Q}_i(X), \widetilde{\mathcal{Q}}_i(X), \text{DP}_i)$ where DP_i is the set of n_i data packets to be authenticated. In order to get the corresponding output, the signature is obtained by querying Sign_{SK} as a black-box at Step 3 of Authenticator.

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

-   There exists a query value i such as:

    $$(\text{BID}, f, n, \alpha, \beta, \mathcal{P}, \rho, \mathcal{Q}(X), \widetilde{\mathcal{Q}}(X)) = (\text{BID}_i, f_i, n_i, \alpha_i, \beta_i, \mathcal{P}_i, \rho_i, \mathcal{Q}_i(X), \widetilde{\mathcal{Q}}_i(X))$$

    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 3 of Authenticator.

-   $|\text{RP} \cap \text{AP}| \geq \lceil \alpha\, n \rceil$ and $|\text{RP}| \leq \lfloor \beta\, n \rfloor$.

-   $\{P'_1, \ldots, P'_n\} = \text{Decoder}(\text{PK}, \text{BID}, f, n, \alpha, \beta, \mathcal{P}, \rho, \mathcal{Q}(X), \widetilde{\mathcal{Q}}(X), \text{RP})$ where $P'_\zeta \neq P_\zeta$ for some $\zeta \in \{1, \ldots, n\}$.

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

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 $r_1 \| \cdots \| r_f \| \sigma$ corresponding to DP once the pad is removed. Note that the length of the pad is uniquely determined once $\mathcal{H}, \mathcal{S}, n$ and ρ are known. The first two ones are general parameters while the others correspond to query i on DP.

The presence of $r_1 \| \cdots \| r_f \| \sigma$ within the list returned by MPR involves that at least one pair message/signature will go through the verification process at Step 3 of Decoder. As the digital signature is unforgeable and the hash function is collision resistant, this pair will be the only one for which $\text{Verify}_{\text{PK}}$ ends successfully. Indeed denote $\widetilde{R} \| \widetilde{\sigma}$ an element from the list such that:

$$\text{Verify}_{\text{PK}}(h(\text{BID} \| f \| n \| \alpha \| \beta \| \mathcal{P} \| \widetilde{R}), \widetilde{\sigma}) = \text{TRUE}$$

By hypothesis, $\mathcal{O}$ is allowed to perform a polynomial number of queries to Authenticator and no more than one query per block identification value. Denote ℓ the number of queries done by $\mathcal{O}$, $\text{BID}_1, \ldots, \text{BID}_\ell$ the ℓ block identification values and $R_1 \| \sigma_1, \ldots,$ $R_\ell \| \sigma_\ell$ the corresponding ℓ concatenations of tree roots/signatures. Note that we are currently working with iteration number i since $\text{BID} = \text{BID}_i$.

Since the signature scheme is secure we get $\widetilde{\sigma} \in \{\sigma_1, \ldots, \sigma_\ell\}$. This means: $\exists\, i_0 \in \{1, \ldots, \ell\}/\widetilde{\sigma} = \sigma_{i_0}$. The security of the digital signature involves $i_0 = i$ as $\mathcal{O}$ cannot query Authenticator more than once per block identification value. Thus: $\widetilde{\sigma} = \sigma_i = \sigma$. For the same reason we get:

$$h(\text{BID} \| f \| n \| \alpha \| \beta \| \mathcal{P} \| \underbrace{R_i}_{R}) = h(\text{BID} \| f \| n \| \alpha \| \beta \| \mathcal{P} \| \widetilde{R})$$

Since h is collision resistant we get: $\widetilde{R}\|\widetilde{\sigma} = R\|\sigma$ which corresponds to the data packets DP ($=$ DP$_i$).

Therefore at the end of Step 3 we have recovered the f tree roots, that is:

$$\forall i \in \{1,\ldots,f\} \quad r'_i = r_i$$

Since h is collision resistant, it is obvious that, for any element of RP written as BID$\|j_i\|C'_{j_i}\|A_{j_i}\|$ path$'_{j_i}$, if path$'_{j_i}$ can be used to recover the path of $h(C'_{j_i})$ to the root of his tree $r'_{\ell_{j_i}} = r_{\ell_{j_i}}$ then $C'_{j_i} = C_{j_i}$. This corresponds to the use of the Merkle hash trees as collision resistant accumulators as in [15]. This involves that, at the end of Step 4 of Decoder, we have:

$$\forall \xi \in \{1,\ldots,n\} \quad C'_\xi \in \{\emptyset, C_\xi\} \quad \text{where } C' = (C'_1 \cdots C'_n)$$

Since $|RP \cap AP| \geq \lceil \alpha\, n \rceil$, we deduce that at least $\lceil \alpha\, n \rceil$ coordinates of C' are non-empty at the end of Step 4. Since the code can correct up to $n - \lceil \alpha\, n \rceil$ erasures we get:

$$\forall \xi \in \{1,\ldots,\lceil \alpha\, n \rceil\} \quad M'_\xi = M_\xi$$

at the end of Step 5.1. Therefore we get:

$$\forall \xi \in \{1,\ldots,n\} \quad P'_\xi = P_\xi$$

We obtain a contradiction with our original hypothesis which stipulated $\exists j \in \{1,\ldots,n\}\, P'_j \neq P_j$. As a consequence, we deduce that either the hash function is not collision resistant or the digital signature is not secure.

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

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

I. Authenticator was never queried on input BID, $f, n, \alpha, \beta, \mathcal{P}, \rho, \mathcal{Q}(X), \widetilde{\mathcal{Q}}(X)$ and the decoding algorithm Decoder does not reject RP, i.e. $\{P'_1,\ldots,P'_n\} \neq \emptyset$ where $\{P'_1, \ldots,P'_n\} = $ Decoder(BID, $f, n, \alpha, \beta, \mathcal{P}, \rho, \mathcal{Q}(X), \widetilde{\mathcal{Q}}(X), $RP).
II. Authenticator was queried on input BID, $f, n, \alpha, \beta, \mathcal{P}, \rho, \mathcal{Q}(X), \widetilde{\mathcal{Q}}(X)$ for some data packets 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. In particular it has been found a pair $(h(\text{BID}\|f\|n\|\alpha\|\beta\|\mathcal{P}\|R), \sigma)$ (after removing the pad) such that:

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

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

If $\mathcal{O}$ queried Authenticator for block tag BID then denote $(\text{BID}, f_i, n_i, \alpha_i, \beta_i, \mathcal{P}_i, \rho_i,$ $\mathcal{Q}_i(X), \widetilde{\mathcal{Q}}_i(X))$ his query. By hypothesis we have:

$$(\text{BID}, f_i, n_i, \alpha_i, \beta_i, \mathcal{P}_i, \rho_i, \mathcal{Q}_i(X), \widetilde{\mathcal{Q}}_i(X)) \neq (\text{BID}, f, n, \alpha, \beta, \mathcal{P}, \rho, \mathcal{Q}(X), \widetilde{\mathcal{Q}}(X))$$

As said in Sect. 3, when $(n, f, \alpha, \beta, \mathcal{P})$ are given, $(\rho, \mathcal{Q}(X), G, \widetilde{\mathcal{Q}}(X))$ are uniquely determined. Thus the previous relation is equivalent to:

$$(\text{BID}, f_i, n_i, \alpha_i, \beta_i, \mathcal{P}_i) \neq (\text{BID}, f, n, \alpha, \beta, \mathcal{P})$$

Therefore either the previous pair message/signature is a forgery of the signature scheme or the pair $(\text{BID}\|f_i\|n_i\|\alpha_i\|\beta_i\|\mathcal{P}_i\|R_i, \text{BID}\|f\|n\|\alpha\|\beta\|\mathcal{P}\|R)$ is a collision for the hash function f.

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

B   Proof of Theorem 2

Let BID be fixed. Due to the accuracy of (α, β), we could demonstrate as in [39] that, at the end of Step 3 of Decoder, the receiver has recovered the signature σ as well as the f tree roots $r_1, \ldots, r_f$. Similarly to Wong and Lam's and Karlof et al.'s approaches [15, 41] which both relies on a Merkle hash tree construction, Step 4 enables us to identify all correct codeword coordinates amongst the set of received elements since h is a collision resistant hash function. Due to the accuracy of (α, β), we have at least $\lceil \alpha n \rceil$ values which are consistent with $(C_1 \cdots C_n)$. Thus Step 5 successfully ends since the code can correct up to $n - \lceil \alpha n \rceil$ erasures. As a consequence, Decoder outputs the whole n original packets, that is: $\forall i \in \{1, \ldots, n\} \, P_i' = P_i$.

As we use the same settings as in [39], we deduce that $U(n) \in O(1)$ is also a bound on the size of the list output by Poly-Reconstruct for our construction. The reader interested in the details is referred to [39].