

Combining Prediction Hashing and MDS Codes for Efficient Multicast Stream Authentication*

Christophe Tartary¹ and Huaxiong Wang^{1,2}

¹Centre for Advanced Computing, Algorithms and Cryptography
Department of Computing
Macquarie University
NSW 2109 Australia

²Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University
Singapore

ctartary@ics.mq.edu.au
HXWang@ntu.edu.sg

Abstract

We study the multicast stream authentication problem when the communication channel is under control of an opponent who can drop, reorder and inject data packets. In this work, we consider that the stream to be authenticated is divided into block of n packets and we assume that the sender can memorize λ such blocks. Two important parameters for stream authentication protocols are packet overhead and computing efficiency. Our construction will exhibit the following advantages. First, our packet overhead will be a few hashes long. Second, the number of signature verifications per family of λ blocks will be $O(1)$ as a function of both λ and n . Third, hash chains will enable the receiver to check the validity of received elements upon reception. As a consequence he will only buffer those consistent with the original data packets. Fourth, the receiver will be able to recover all the data packets emitted by the sender despite erasures and injections by running the decoding algorithm of the maximal distance separable code onto the elements which have passed the previous filtering process.

Keywords: Stream Authentication, Polynomial Reconstruction, Adversarial Network, Erasure Codes, Prediction Hashing, Hash Chains.

1 Introduction

Multicast communication enables a single sender to distribute digital content to a large audience via a public channel such as the Internet. It has applications in sensor networks, pay-TV, air traffic control, stock quotes and military defense systems for instance. Nevertheless, large-scale broadcasts prevent lost content from being redistributed since the lost of any piece of data could generate a prohibitive number of redistribution requests at the sender. Furthermore, the channel can be under the control of adversaries performing malicious actions on the data stream¹. Thus, the security of multicast protocols relies on two aspects: the opponents' computational powers and the network properties. Several unconditionally secure schemes were designed in [4, 10, 41]. Unfortunately, their optimal security is at the cost of a large storage requirement or a single-time use which makes these constructions unsuitable for practical applications. In this paper, we will assume that the adversaries have bounded computational powers.

In recent years, many protocols were designed to deal with the multicast authentication problem [5]. An application such as a TV channel broadcasting 24 hours a day implies that the size of the stream can be infinite. On the other hand, the receivers must be able to authenticate data within a short period of delay upon reception. Since many protocols will distribute private or sensitive content, non-repudiation of the sender is required for most of them as using data from an incorrect origin can have disastrous consequences during military operations for instance. As a consequence, schemes like TESLA [34, 36] and its variations [21, 22, 23, 35, 47] are not suitable since data authenticity is guaranteed using message authentication codes whose keys are disclosed after some period of time. Notice that the assumptions made by Perrig *et al.* to guarantee the security of TESLA were proved to be insufficient by Jakimoski [17]. Thus, constructions for multicast

*The original version of this paper appears in the proceedings of the 12th Australasian Conference on Information Security and Privacy (ACISP 2007), Lecture Notes in Computer Science, vol. 4586, pp 293 - 307, Springer - Verlag.

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

distribution rely on digital signatures to provide non-repudiation. Nevertheless, signing each data packet² is not a practical solution as digital signatures are generally too expensive to generate and/or verify. In addition, bandwidth limitations prevent one-time and k -time signatures [13, 40] from being used due to their size. That is why a general approach consists of generating a single signature and amortizing its computational cost and overhead over several data packets using hash functions for instance.

In order to deal with erasures, Perrig *et al.* [34, 36], Challal *et al.* [6], Golle and Modadugu [14] and Miner and Staddon [28] appended the hash of each packet to several followers according to specific patterns. In these papers, packet loss was modeled by k -state Markov chains [12, 33, 49] and bounds on the packet authentication probability were computed. Nevertheless, the drawback of these schemes is that they rely on the reception of signed packets which cannot be guaranteed over networks such as the Internet since they only provide a best effort delivery. This problem restricts the range of applications for the previous protocols.

An approach to overcome this problem is to split the signature into k smaller parts where only ℓ of them ($\ell < k$) are sufficient for recovery. Different techniques were employed to obtain the dispersion of the signature: Perrig *et al.* [30, 31] and Park and Cho [32] used the Information Dispersal Algorithm [38], Al-Ibrahim and Pieprzyk [1] combined linear equations and polynomial interpolation whereas Pannetrat and Molva [29] utilized erasure codes. Unfortunately, none of these constructions tolerates a single packet injection which is a major drawback since it is unlikely to have only reliable network nodes between the sender and each receiver if you consider, in particular, the Internet.

In 2003, Lysyanskaya *et al.* [24] used Reed-Solomon codes [39] to design a protocol resistant to packet loss and data injections. Their augmented packets³ are $O(1)$ bits long while the number of signature verifications per block⁴ turns out to be $O(1)$ as functions of the block length n . In 2004, Karlof *et al.* designed a scheme called PRABS [18] combining an erasure code (to recover lost content) and a one-way accumulator [2, 3] based on a Merkle hash tree [27] (to deal with injections). This approach is similar to Wong and Lam's scheme [48] but the number of signature verifications for PRABS is $O(1)$ even in the worst case. The bound on the number of signature verifications for PRABS is much smaller than in [24] (see [46]) but this is at the expense of having $\Theta(\log_2(n))$ -bit augmented packets since each of them has to carry $\lceil \log_2(n) \rceil$ hashes.

In order to reduce this overhead, Di Pietro *et al.* proposed a modified distribution of hashes so that the Merkle hash tree can still be reconstructed [11]. Another benefit of their scheme is to decrease the number of decoding operations to be performed at the receiver. Nevertheless this approach has two drawbacks. First, some augmented packets still carry $\lceil \log_2(n) \rceil$ hashes while others only have a few digests. This results in important variations in packet sizes leading to irregular throughput of information in the channel and can cause data congestion in the network. Second, the number of signature verifications to be performed by the receiver is equal to the number of injections in the worst case which creates a potential weakness against Denial-of-Service (DoS) attacks. In [7], Choi used PRABS as a subroutine to ensure the security of his prediction hashing-based construction. Nevertheless, this scheme exhibits the same logarithmic overhead as PRABS. Recently, Tartary and Wang proposed a construction based on [24] and Maximal Distance Separable (MDS) codes [25] (denoted TWMDS in this paper) which is resistant against packet loss and data injections and requires $O(1)$ signature verifications like PRABS but only has a $O(1)$ -bit packet overhead and allows recovery of all data packets [46].

As the number of signature verifications for TWMDS is higher than PRABS (see Table 2), we propose a multiple block construction similar to the approach by Tartary and Wang [45] (denoted TWMB in this article). As TWMB, we will generate a single signature per family of λ blocks where each of them consists of n packets. The receivers will still be able to authenticate data per block and it is possible to join the communication group at any block boundary as in [45]. The number of signature verifications per family of λ block will be identical to the number of verifications for a single block of TWMDS. As for TWMDS, we will use MDS codes to provide full recovery of data packets. The security of schemes in [24, 45, 46] relies on the use of a polynomial time algorithm by Guruswami and Sudan called Poly-Reconstruct [16]. The idea of [prediction hashing \(PH\)](#) is that each block of n packets conveys information which will be used to authenticate (or predict) the following block of packets. Using PH, our construction will enable to filter elements upon reception and thus the receiver will exclusively buffer elements consistent with the original data stream. The first advantage is that memory is not wasted by storing irrelevant pieces of data contrary to [7, 18, 24, 45, 46]. The second benefit is that the previous filtering process will also reduce the number of queries to Poly-Reconstruct to 2 per family of λ blocks which will speed up the authentication process at the receiver considerably. The authenticated packets of our construction will still be $O(1)$ bits long as for TWMB and TWMDS.

This paper is organized as follows. In the next section, we will present our network model as well as an algorithm from [46] to be used as a subroutine in our construction. In Sect. 3, we will describe our authentication scheme. Its security and

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

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

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

recovery property will be studied in Sect. 4. In Sect. 5, we will compare our scheme to PRABS, TWMB and TWMDs as our work can be seen as their extension. Finally, we will summarize our contribution to the multicast authentication problem.

2 Preliminaries

In this section, we introduce the assumptions and constructions to be used as subroutines for our scheme. First, we present our network model. Second, we justify our choice of erasure codes. Finally, we recall a modified version of the algorithm Poly-Reconstruct by Guruswami and Sudan since it will play a key role to deal with packet injections as in [24, 45, 46].

2.1 Network Model

We consider that the communication channel is unsecured. This means that it is under the control of an opponent \mathcal{O} who can drop and rearrange packets of his choice as well as inject bogus data into the network [26]. Our area of investigation is 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 elements are injected by \mathcal{O} . Indeed, if too many original packets are dropped then data transmission becomes the main problem to treat since the small number of received elements would be probably useless even authenticated. On this other hand, if \mathcal{O} injects a large number of forged packets then the main problem to be solved becomes increasing the resistance against DoS attacks. In order to build our signature amortization scheme, we need to split the data stream into blocks of n packets: P_1, \dots, P_n . We define two parameters: α ($0 < \alpha \leq 1$) (the *survival* rate) and β ($\beta \geq 1$) (the *flood* rate). It is assumed that at least a fraction α and no more than a multiple β of the number of augmented packets are received. This means that at least $\lceil \alpha n \rceil$ original augmented packets are received amongst a total which does not exceed $\lfloor \beta n \rfloor$ elements.

We would like to point out that we are not interested in the cases ($\alpha = 1$) and ($\beta = 1$). Indeed, in the first case, all original data packets are received. Thus we only need to distinguish correct elements from bogus ones which can be achieved using Wong and Lam's technique [48]. In the second case, there are no packet injections from \mathcal{O} . Thus, using an erasure code (see [9] as an example) is sufficient to recover P_1, \dots, P_n . Therefore, in this work, we will only study the case: $0 < \alpha < 1 < \beta$. Notice, however, that our construction also works when $\alpha = 1$ and $\beta = 1$.

2.2 Code Construction

In this paper, we consider linear codes. A linear code of length N , dimension K and minimum distance D is denoted $[N, K, D]$. The Singleton bound states that any $[N, K, D]$ code satisfies: $D - 1 \leq N - K$ [25]. It is known that any $[N, K, D]$ code can correct up to $D - 1$ erasures [50]. Thus, a $[N, K, D]$ code cannot correct more than $N - K$ erasures. In order to maximize the efficiency of our construction, we are interested in codes correcting exactly $N - K$ erasures. These codes are called *Maximum Distance Separable (MDS)* codes [25]. Even if the scheme we propose works with any MDS code, we suggest to use the construction by Lacan and Fimes [19] for better practical efficiency (see [46] for details). 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: $c := mG$ [25].

2.3 Polynomial Reconstruction Algorithm

In [16], 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 modified version of Poly-Reconstruct as in [46] 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 [20]. Operations in \mathbb{F}_{2^q} are performed modulo a polynomial $Q(X)$ of degree q which is irreducible over \mathbb{F}_2 .

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 [15]).

3 Our Construction

We need a collision resistant hash function h [37] and an unforgeable signature scheme $(\text{Sign}_{\text{SK}}, \text{Verify}_{\text{PK}})$ [43] the key pair of which (SK, PK) is created by a generator *KeyGen* as in [18, 24, 45, 46].

Algorithm 1 MPR

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

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

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

3.1 Scheme Overview

We have λ blocks of packets $\{P_{i,1}, \dots, P_{i,n}\}_{i=1, \dots, \lambda}$. In order to use PH, we proceed backwards. We encode the last block using the $[n, \lceil \alpha n \rceil, n - \lceil \alpha n \rceil + 1]$ code into the codeword $(C_{\lambda,1}, \dots, C_{\lambda,n})$. Then, we append the hashes $h(C_{\lambda,1}), \dots, h(C_{\lambda,n})$ to the packets of block $\lambda - 1$ and encode the resulting n elements into $(C_{\lambda-1,1}, \dots, C_{\lambda-1,n})$. We repeat this process to the first block of packets. We generate the family signature as in [45]. That is, we compute the λ block hashes $h_i := h(h(C_{i,1}) \parallel \dots \parallel h(C_{i,n}))$ and sign $h(h_1 \parallel \dots \parallel h_\lambda)$ into σ . We build the family polynomial $\mathcal{F}(X)$ of degree at most ρn (for some constant ρ) the coefficients of which represent $h_1 \parallel \dots \parallel h_n \parallel \sigma$. In order to allow new members to join the communication group at block boundaries, we build λ block polynomials $\mathcal{B}_1(X), \dots, \mathcal{B}_\lambda(X)$ of degree at most ρn such as the coefficients of each $\mathcal{B}_i(X)$ represent $h(C_{i,1}) \parallel \dots \parallel h(C_{i,n})$. The augmented packets of the family of λ blocks are such as:

$$\forall i \in \{1, \dots, \lambda\} \forall j \in \{1, \dots, n\} \text{AP}_{i,j} := \text{FID} \parallel i \parallel j \parallel C_{i,j} \parallel \mathcal{B}_i(j) \parallel \mathcal{F}(j)$$

where FID represents the position of the family $P_{1,1}, \dots, P_{\lambda,n}$ within the whole stream.

Upon reception of data for the i^{th} block, the receiver adapts his reaction whether or not he knows its digests.

- If the hashes are known (via PH) then he only needs to filter the received elements and drop those which are inconsistent with those digests. Finally, he corrects erasures using the MDS code to recover the n data packets $\{P_{i,1}, \dots, P_{i,n}\}$ as well as the n hashes corresponding to block $i + 1$ which updates the values for PH.
- If the hashes are unknown then he proceeds as in [45]. That is, he first checks whether the family signature corresponding to data he obtained is valid by reconstructing $\mathcal{F}(X)$. If so, he checks whether the block information is consistent with the previous signature by reconstructing $\mathcal{B}_i(X)$. Then, he sorts the received pieces of data and drops those which are inconsistent with $\mathcal{B}_i(X)$. Finally, he corrects erasures using the MDS code to recover the n data packets $\{P_{i,1}, \dots, P_{i,n}\}$ as well as the n hashes corresponding to block $i + 1$ which updates the values for PH.

3.2 Formal Scheme Construction

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 [46] 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. The values of q, \tilde{q} as well as the length of the different pads used by our scheme can be found in Appendix A. Table 1 summarizes the scheme parameters which are assumed to be publicly known.

n : Block length	$\tilde{Q}(X)$: Polynomial representing the field for the MDS code
λ : Family length	\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

Table 1: Public parameters for our authentication scheme.

The hash function h as well as Verify and PK are also assumed to be publicly known. We did not include them in Table 1 since they can be considered as general parameters. For instance, h can be SHA-256 while the digital signature is a 1024-bit RSA signature. We denote \mathcal{H} the digest bit length and s the bit length of a signature. Since h and the digital signature are publicly known, so are \mathcal{H} and s .

Algorithm 2 Authenticator

Input: The family number FID, the secret key SK, the parameters of Table 1 and data packets $P_{1,1}, \dots, P_{\lambda,n}$.

/* Packet Encoding */

1. Parse $P_{\lambda,1} \parallel \dots \parallel P_{\lambda,n}$ as $M_{\lambda,1} \parallel \dots \parallel M_{\lambda, \lceil \alpha n \rceil}$ after padding. Encode the message $(M_{\lambda,1}, \dots, M_{\lambda, \lceil \alpha n \rceil})$ into the codeword $(C_{\lambda,1}, \dots, C_{\lambda,n})$ using the MDS code.
2. For i from $\lambda - 1$ to 1 do
 - 2.1. Compute the hashes $h(C_{i+1,j})$ for $j \in \{1, \dots, n\}$ and append them to packets of block i as: $\tilde{P}_{i,j} := P_{i,j} \parallel h(C_{i+1,j})$.
 - 2.2. Parse $\tilde{P}_{i,1} \parallel \dots \parallel \tilde{P}_{i,n}$ as $M_{i,1} \parallel \dots \parallel M_{i, \lceil \alpha n \rceil}$ after padding. Encode the message $(M_{i,1}, \dots, M_{i, \lceil \alpha n \rceil})$ into the codeword $(C_{i,1}, \dots, C_{i,n})$ using the MDS code.

/* Block Identification */

3. For i from 1 to λ do
 - 3.1. Parse $h(C_{1,1}) \parallel \dots \parallel h(C_{i,n})$ as $b_{i,0} \parallel \dots \parallel b_{i,\rho n}$ where each $b_i \in \mathbb{F}_{2^q}$ after padding and compute the block hash h_i as $h_i := h(h(C_{1,1}) \parallel \dots \parallel h(C_{i,n}))$.
 - 3.2. Construct the block polynomial $\mathcal{B}_i(X) := b_{i,0} + b_{i,1} X + \dots + b_{i,\rho n} X^{\rho n}$ and evaluate it at the first n points of \mathbb{F}_{2^q} .

/* Signature Generation */

4. Write h_f as $h_f := h_1 \parallel \dots \parallel h_\lambda$. Compute the family signature σ as $\sigma := \text{Sign}_{\text{SK}}(h(\text{FID} \parallel \lambda \parallel n \parallel \alpha \parallel \beta \parallel \rho \parallel h_f))$. Parse $h_f \parallel \sigma$ as $f_0 \parallel \dots \parallel f_{\rho n}$ where each $f_i \in \mathbb{F}_{2^q}$ after padding.
5. Construct the family polynomial $\mathcal{F}(X) := f_0 + f_1 X + \dots + f_{\rho n} X^{\rho n}$ and evaluate it at the first n points of \mathbb{F}_{2^q} .

/* Construction of Augmented Packets */

6. Build the augmented packet $\text{AP}_{i,j}$ as $\text{AP}_{i,j} := \text{FID} \parallel i \parallel j \parallel C_{i,j} \parallel \mathcal{B}_i(j) \parallel \mathcal{F}(j)$ for $i \in \{1, \dots, \lambda\}$ and for $j \in \{1, \dots, n\}$.

Output: The λn augmented packets $\{\text{AP}_{1,1}, \dots, \text{AP}_{\lambda,n}\}$ which are sent to the network per block of n elements $\{\text{AP}_{i,1}, \dots, \text{AP}_{i,n}\}_{i=1, \dots, \lambda}$.

Notice that the list of irreducible polynomials over \mathbb{F}_2 is used at Step 5 of Algorithm 2 when performing polynomial evaluations over \mathbb{F}_{2^q} . Those evaluations work as follows. Since any element of \mathbb{F}_{2^q} can be represented as $\lambda_0 Y^0 + \lambda_1 Y^1 + \dots + \lambda_{q-1} Y^{q-1}$ where each λ_i belongs to \mathbb{F}_2 , we define the first n elements as $(0, \dots, 0)$, $(1, 0, \dots, 0)$, $(0, 1, 0, \dots, 0)$, $(1, 1, 0, \dots, 0)$ and so on until the binary decomposition of $n - 1$.

We depict our decoding algorithm in Fig 1. It uses as subroutines FilterElements and DecoderBlock. The latter is the decoding algorithm from TWMB. In our construction, however, it will be queried once per family of λ blocks while FilterElements will be run $\lambda - 1$ times. This will create a speed-up at the receiver as it only performs hash computations and erasure corrections as well as memory savings since the buffered elements will be consistent with the original codewords (see proof of Theorem 2).

In order to verify the correctness of the family signature, the receiver will use the same algorithm VerifySignatureFamily as TWMD [45].

Algorithm 3 VerifySignatureFamily

Input: The family number FID, the public key PK, the elements of Table 1 and a set of pairs of field elements $\{(x_i, y_i), 1 \leq i \leq m\}$.

1. Run MPR on $\{(x_i, y_i), 1 \leq i \leq m\}$ to get a list \mathcal{L} of candidates for the family signature verification. If MPR rejects this input then the algorithm stops.
2. While the signature has not been verified and the list \mathcal{L} has not been exhausted, pick a new candidate $\tilde{h}_1 \parallel \dots \parallel \tilde{h}_\lambda \parallel \tilde{\sigma}$. If $\text{Verify}_{\text{PK}}(h(\text{FID} \parallel \lambda \parallel n \parallel \alpha \parallel \beta \parallel \rho \parallel \tilde{h}_1 \parallel \dots \parallel \tilde{h}_\lambda), \tilde{\sigma}) = \text{TRUE}$ then $\tilde{\sigma}$ is considered as the authentic family signature σ and the \tilde{h}_i 's are memorized within the table HashBlock as the authentic block digests h_i 's.
3. If the signature has not been verified then our algorithm stops.

Output: $(\sigma, \text{HashBlock})$: family signature and hashes of the λ blocks.

Our scheme embeds the digests of the codeword related to block $i + 1$ into block i . This will enable each receiver

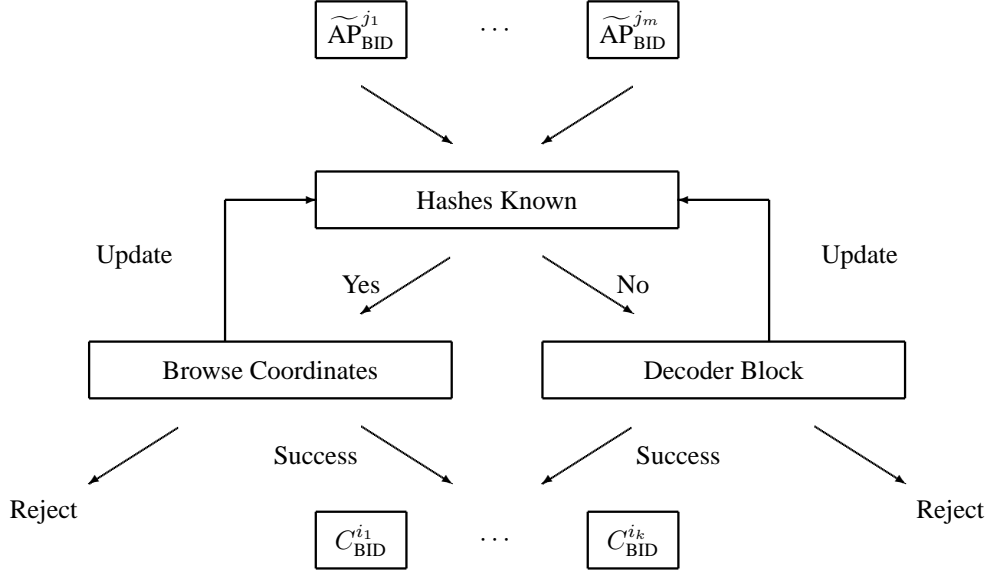


Figure 1: A sketch of our decoding algorithm DynamicDecoder for block BID.

to filter data in order to speed up the authentication scheme and reduce the number of elements to be buffered. We now present FilterElements which provides on-the-fly verification of received elements.

It should be noticed that the boolean value KnownHashes indicates if the table of digests HashCodeword has been updated. This enables the receiver to switch between buffering all incoming data elements and on-the-fly validation of data.

The reader may notice that we only verified the consistency of the substring $\text{FID}_i \parallel \text{BID}_i \parallel j_i \parallel C'_{\text{BID}_i, j_i}$ to our parameters FID, BID, n and HashCodeword. Since we did not check any condition on $\mathcal{B}_{\text{BID}_i, j_i} \parallel \mathcal{F}_{\text{BID}_i, j_i}$, one may think that an opponent can submit an incorrect element making our decoding process fail. We would like to emphasize that it is not the case. As just noticed, the elements going successfully through this process are written as $\text{FID} \parallel \text{BID} \parallel \theta \parallel C_{\text{BID}, \theta} \parallel x \parallel y$ for some $\theta \in \{1, \dots, n\}$. Nevertheless, the substring $x \parallel y$ does not play any role in our algorithm since we only use $C_{\text{BID}, \theta}$ to recover the original data packets. Therefore, even if $x \parallel y$ is a bogus string (i.e. $x \parallel y \neq \mathcal{B}_{\text{BID}}(\theta) \parallel \mathcal{F}(\theta)$) then it has no influence whatsoever on the output of FilterElements which makes the attack by the adversary pointless.

The array T is used to dodge duplication attacks by an opponent who would submit several strings $\text{FID} \parallel \text{BID} \parallel \theta \parallel C_{\text{BID}, \theta} \parallel x \parallel y$ (for different values of $x \parallel y$) in order to exhaust the receiver computational power by recomputing $h(C_{\text{BID}, \theta})$ whereas the original coordinate $C_{\text{BID}, \theta}$ has already been recovered.

We now introduce DecoderBlock used for the first block of the family. Notice that DecoderBlock is a modification of DecoderBlock $_{\epsilon}$ from [45].

Finally, we build our dynamic decoder run by the receivers to authenticate data. We assume that the boolean value KnownHashes is set to FALSE when a receiver joins the communication group and re-initialized to FALSE when the receiver processes the first received block of a new family $\text{FID}' (> \text{FID})$.

Algorithm 6 DynamicDecoder

Input: The family number FID, the block number BID, the public key PK, Table 1, a boolean KnownHashes, a table HashCodeword and a set of received packets RP.

If KnownHashes = FALSE then

Query DecoderBlock on input (PK, FID, BID, $\lambda, n, \alpha, \beta, \rho, \mathcal{Q}(X)$, RP).

Else

Query FilterElements on input (FID, BID, $\lambda, n, \alpha, \beta, \rho$, HashCodeword, RP).

Output: The set of identified packets $\{P'_{\text{BID}, 1}, \dots, P'_{\text{BID}, n}\}$, the updated boolean value KnownHashes and the updated table HashCodeword as output of either DecoderBlock or FilterElements.

Note that when DynamicDecoder stops then the whole content of block BID is lost. Nevertheless, the definitions of

Algorithm 4 FilterElements

Input: The family number FID, the block number BID, the elements of Table 1, a table HashCodeword and a flow of packets.

1. Set $T(i) := 0$ for $i \in \{1, \dots, n\}$, set $\mathcal{C}' := (\emptyset, \dots, \emptyset)$ and $\text{KnownHashes} = \text{FALSE}$.
2. Upon reception of a new data packet do
 - 2.1. Write it as $\text{FID}_i \parallel \text{BID}_i \parallel j_i \parallel \mathcal{C}'_{\text{BID}_i, j_i} \parallel \mathcal{B}_{\text{BID}_i, j_i} \parallel \mathcal{F}_{\text{BID}_i, j_i}$. If $\text{FID}_i \neq \text{FID}$ or $\text{BID}_i \neq \text{BID}$ or $j_i \notin \{1, \dots, n\}$ or $T(j_i) = 1$ then discard the packet.
 - 2.2 If $h(\mathcal{C}'_{\text{BID}_i, j_i}) = \text{HashCodeword}(j_i)$ then set $T(j_i) = 1$ and set the j_i^{th} coordinate of \mathcal{C}' to $\mathcal{C}'_{\text{BID}_i, j_i}$.

/* After Reception of all Packets for Values (FID, BID) */

3. If \mathcal{C}' has less than $\lceil \alpha n \rceil$ non-erased coordinates then the algorithm stops.

Else

3.1. Correct the erasures of \mathcal{C}' using the MDS decoding process and denote $(M'_{\text{BID},1}, \dots, M'_{\text{BID}, \lceil \alpha n \rceil})$ the corresponding message.

3.2. Remove the pad from $M'_{\text{BID},1} \parallel \dots \parallel M'_{\text{BID}, \lceil \alpha n \rceil}$ and write the resulting string as

$$\begin{cases} P'_{\text{BID},1} \parallel h'_{\text{BID},1} \parallel \dots \parallel P'_{\text{BID},n} \parallel h'_{\text{BID},n} & \text{if } \text{BID} \neq \lambda \\ P'_{\text{BID},1} \parallel \dots \parallel P'_{\text{BID},n} & \text{otherwise} \end{cases}$$

where each $P'_{\text{BID},i}$ is \mathcal{P} bits long and each $h'_{\text{BID},i}$ is \mathcal{H} bits long.

3.3. If $\text{BID} \neq \lambda$ then set $\text{HashCodeword}(i) = h'_{\text{BID},i}$ for $i \in \{1, \dots, n\}$ and set $\text{KnownHashes} = \text{TRUE}$.

Output: The set of identified packets $\{P'_{\text{BID},1}, \dots, P'_{\text{BID},n}\}$, the boolean value KnownHashes and HashCodeword containing the digests of the next block.

α and β ensure that this will never happen (see Theorem 2). In a practical point of view, one can choose α small and β large enough so that the real threat of the opponent is bounded by those values. Nevertheless, inaccurate values, such as $\alpha = 10^{-10}$ and $\beta = 10^{10}$ for instance, will lead to excessive overhead and computation. So, the values α and β set by the sender should accurately reflect the opponent actual ability. Developing techniques allowing the determination of such values is beyond the scope of this paper.

4 Security and Recovery Analysis

4.1 Security of the Scheme

We adopt the same security definition as in [45]. It can be seen as an extension to the notion of "family of blocks" of the definitions from [24, 46]. The definition is as follows:

Definition 1 ($\text{KeyGen}, \text{Authenticator}, \text{DynamicDecoder}$) 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 family identification tag FID).
- iii. \mathcal{O} outputs $(\text{FID}, \text{BID}, \lambda, n, \alpha, \beta, \rho, \mathcal{Q}(X), \text{RP})$.

\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 $\lfloor \beta n \rfloor$) of some authenticated packets set AP for some family identification tag FID and block identification tag BID , the decoder still fails at identifying some of the correct packets.
- b. (security violation) \mathcal{O} succeeds to output RP such that the decoder returns $\{P'_{\text{BID},1}, \dots, P'_{\text{BID},n}\}$ (for some $\text{BID} \in \{1, \dots, \lambda\}$) that was never authenticated by Authenticator (as the BID^{th} block of a family of λ blocks) for the family tag BID and parameters $(\lambda, n, \alpha, \beta, \rho, \mathcal{Q}(X))$.

Algorithm 5 DecoderBlock

Input: The family number FID, the block number BID, the elements of Table 1, a set of received packets RP.

/* Signature Verification */

1. Write the packets as $\text{FID}_i \parallel \text{BID}_i \parallel j_i \parallel C'_{\text{BID}_i, j_i} \parallel \mathcal{B}_{\text{BID}_i, j_i} \parallel \mathcal{F}_{\text{BID}_i, j_i}$ and discard those having $\text{FID}_i \neq \text{FID}$, $\text{BID}_i \neq \text{BID}$ or $j_i \notin \{1, \dots, n\}$. Denote m' the number of remaining packets. If $m' < \lceil \alpha n \rceil$ or $m' > \lfloor \beta n \rfloor$ then the algorithm stops.
2. Run VerifySignatureFamily on the m' remaining points $\{(j_i, \mathcal{F}_{\text{BID}_i, j_i}), 1 \leq i \leq m'\}$. If it rejects the input then the algorithm stops.
3. Run MPR on the set $\{(j_i, \mathcal{B}_{\text{BID}_i, j_i}), 1 \leq i \leq m'\}$ and get a list L of candidates for block tag verification. If MPR rejects that set then the algorithm stops.

/* Block Hashes Verification */

4. While the hash for block BID has not been verified and the list L has not been exhausted, we pick a new candidate $\tilde{c} := \tilde{h}_{\text{BID}}^1 \parallel \dots \parallel \tilde{h}_{\text{BID}}^n$. If $(h(\tilde{c}) = \text{HashBlock}(\text{BID}))$ then the tag of block BID is verified and we set $h_{\text{BID}}^j = \tilde{h}_{\text{BID}}^j$ for $j \in \{1, \dots, n\}$. If L is exhausted without a successful block tag verification then the algorithm stops.

/* Packet Decoding */

5. Set $C' := (\emptyset, \dots, \emptyset)$ and KnownHashes := FALSE. For each of the m' remaining packets, $\text{FID} \parallel \text{BID} \parallel j_i \parallel C'_{\text{BID}_i, j_i} \parallel \mathcal{B}_{\text{BID}_i, j_i} \parallel \mathcal{F}_{\text{BID}_i, j_i}$, if $h(C'_{\text{BID}_i, j_i}) = h_{\text{BID}}^t$ for some $t \in \{1, \dots, n\}$ then set the t^{th} coordinate of C' to C'_{BID_i, j_i} .
6. Perform Step 3 of FilterElements to recover the data packets as well as the digests of the next block to be stored into HashCodeword.

Output: The set of identified packets $\{P'_{\text{BID},1}, \dots, P'_{\text{BID},n}\}$, the boolean value KnownHashes and HashCodeword containing the digests of the next block.

As in [24, 45, 46], we have the following result regarding the security and correctness of our construction whose proof can be found in Appendix B.

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

4.2 Recovery Property

We now prove that our scheme enables any receiver to recover the n data packets for any of the λ blocks and the number of signature verifications to be performed per family is $O(1)$ as a function of both n and λ . As in [45, 46], we introduce the following definition:

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

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

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

We now assume that (α, β) is accurate for our network flow n in the remaining of this paper. As shown in [46], it is a realistic assumption to consider the accuracy of (α, β) for PRABS as well. We have the following result whose proof can be found in Appendix C.

Theorem 2 *For any FID, for any BID, each receiver recovers the n original data packets $P_{\text{BID},1}, \dots, P_{\text{BID},n}$. In addition, the number of signature verifications to be performed for the whole family of λ blocks is upper bounded by $U(n) := \min(\lfloor U_1(n) \rfloor, \lfloor U_2(n) \rfloor)$ where:*

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

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

5 Comparison of Authentication Protocols

In this section, we will compare our construction to PRABS, TWMB and TWMDS as our approach can be seen as their extension. As underlined in Sect. 1, the computing efficiency and the packet overhead are two important factors to determine the practicality of a stream authentication protocol. Our comparison focuses on these two factors.

5.1 Computing Efficiency

In the proof of Theorem 2 (see the extended version of the article for details), it is shown that DecoderBlock is queried only once for the whole family of λ blocks. Thus, Poly-Reconstruct is run only twice per family (once within VerifySignatureFamily and once at Step 3 of DecoderBlock). At the same time, the receiver can filter elements for the remaining $\lambda - 1$ blocks. Using PH, our filtering process allows efficient buffering and faster authentication as the receiver:

- treats elements upon reception (on-the-fly verification).
- memorizes only the correct code coordinates.

Table 2 summarizes the benefits provided by the different authentication schemes. In order to have a fair comparison, we assumed that PRABS and TWMDS were iterated λ times. Notice that the value $V(n)$ can be found in [18] and is equal to $\left\lceil \frac{\lfloor \beta n \rfloor}{\lfloor \alpha n \rfloor} \right\rceil$. Remark that a comparison between $U(n)$ and $V(n)$ (when $n = 1000$) for different pairs (α, β) can be found in [46].

	Signature Verification	Complexity	Calls to Poly-Reconstruct	Filtering	Total Recovery
Our Scheme	$U(n)$	$O(1)$	2	Yes	Yes
PRABS	$\lambda V(n)$	$O(\lambda)$	N/A	No	No
TWMB	$U(n)$	$O(1)$	$\lambda + 1$	No	No
TWMDS	$\lambda U(n)$	$O(\lambda)$	λ	No	Yes

Table 2: Efficiency comparison for multicast stream protocols.

5.2 Packet Overhead

In our scheme, augmented packets sent through the network are written as: $\text{FID} \| i \| j \| C_{i,j} \| \mathcal{B}_i(j) \| \mathcal{F}(j)$. 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: $\text{FID} \| i \| j \| P_{i,j}$. Remember that the bit size of packets $P_{i,j}$ is \mathcal{P} . Our overhead is:

$$\text{length}(C_{i,j}) + \text{length}(\mathcal{B}_i(j)) + \text{length}(\mathcal{F}(j)) - \mathcal{P}$$

The element $C_{i,j}$ belongs to the field used for the MDS code. Thus it is \tilde{q} bits long. In addition $\mathcal{B}_i(j)$ and $\mathcal{F}(j)$ are q bits long. Using the values q and \tilde{q} (see the extended version of this paper), we deduce that our packet overhead is:

$$\left\lceil \frac{n(\mathcal{P} + \mathcal{H})}{\lfloor \alpha n \rfloor} \right\rceil + 2 \left\lceil \frac{\max(n\mathcal{H}, \lambda\mathcal{H} + s)}{\rho n + 1} \right\rceil - \mathcal{P}$$

which is $O(1)$ as a function of the block length n . Notice that when n is large the previous value can be approximated by:

$$\left(\frac{1}{\alpha} - 1 \right) \mathcal{P} + \left(\frac{1}{\alpha} + \frac{2}{\rho} \right) \mathcal{H}$$

Table 3 summarizes the overhead comparison of the different authentication schemes.

	Bit Size	Complexity
Our Scheme	$\left(\left\lceil \frac{n(\mathcal{P} + \mathcal{H})}{\lfloor \alpha n \rfloor} \right\rceil - \mathcal{P} \right) + 2q$	$O(1)$
PRABS	$\frac{n\mathcal{H} + s}{\lfloor \alpha n \rfloor} + \log_2(n) \mathcal{H}$	$\Theta(\log_2(n))$
TWMB	$2q$	$O(1)$
TWMDS	$\left(\left\lceil \frac{n\mathcal{P}}{\lfloor \alpha n \rfloor} \right\rceil - \mathcal{P} \right) + \frac{n\mathcal{H} + s}{\rho n + 1}$	$O(1)$

Table 3: Overhead comparison for multicast stream protocols.

Notice that the values $\left(\left\lceil \frac{n(\mathcal{P} + \mathcal{H})}{\lfloor \alpha n \rfloor} \right\rceil - \mathcal{P} \right)$ and $\left(\left\lceil \frac{n\mathcal{P}}{\lfloor \alpha n \rfloor} \right\rceil - \mathcal{P} \right)$ correspond to a stretching coefficient roughly equal to $\frac{1}{\alpha} - 1$. This is the price to pay in order to use the MDS code which guarantees total recovery of all data packets. The apparent

low overhead of TWMB comes from the fact that scheme does not provide recovery of lost content (see Table 2). Remark that when the survival rate α is close to 1 (i.e. the channel has a good delivery rate), the previous values get close to \mathcal{H} and 0 respectively. Thus the packet overhead of our construction is asymptotically $\left(1 + \frac{2}{\rho}\right)$ hashes long.

If we compare our construction to TWMDS then our packet overhead is one field element plus, roughly, $\frac{\mathcal{H}}{\alpha}$ bits longer. Notice, however, that our field elements are smaller than those in TWMDS. Indeed λ can be seen as small in comparison to the block length n and thus $q < \left\lceil \frac{n\mathcal{H}+s}{\rho n+1} \right\rceil$.

6 Conclusion

In this paper, we presented a stream authentication protocol which can be considered as an extension of PRABS, TWMB and TWMDS presented in [18, 45, 46]. Our construction only requires the generation of a single signature for every family of λ blocks of n packets and allows any group member to join the communication group at any block boundary as TWMB. The number of signature verifications to be performed at the receiver and the bit size of our packet overhead are $O(1)$ as functions of n and λ , contrary to PRABS, where they are linear in λ and logarithmic in n respectively. These two advantages already existed for TWMB but our construction also allows total recovery of the data packets which is not provided by either TWMB or PRABS. This feature is beneficial when the packets represent audio or video information as our approach prevent audio gaps and frozen images for instance. Furthermore, using PH, we are able to save memory at the receiver as those hash chains enable him to decide whether or not dropping data packets upon reception which is not possible for any of PRABS, TWMB and TWMDS. As a consequence, the running time at the receiver is decreased since he only needs to use the erasure code to recover the data packets contrary to PRABS, TWMB and TWMDS where he must first build Merkle hash trees (for PRABS) or query Poly-Reconstruct (for TWMB and TWMDS) before using the erasure code.

It should be noticed that we can improve the running time at the receiver even further by using a trapdoor hash function [42] (such as Very Smooth Hash [8] for instance) instead of a digital signature as in [44].

Acknowledgment

The authors would like to thank Professor Josef Pieprzyk for his valuable comments on this work. The authors are also grateful to the anonymous reviewers for their feedback to improve the quality of this paper. This work was supported by the Australian Research Council under ARC Discovery Projects DP0558773 and DP0665035. The first author's work was also funded by an iMURS scholarship supported by Macquarie University.

References

- [1] Mohamed Al-Ibrahim and Josef Pieprzyk. Authenticating multicast streams in lossy channels using threshold techniques. In *ICN 2001*, volume 2094 of *Lecture Notes in Computer Science*, pages 239 – 249, Colmar - France, July 2001. Springer - Verlag.
- [2] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology - Eurocrypt'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480 – 494, Konstanz - Germany, May 1997. Springer - Verlag.
- [3] 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.
- [4] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kuten, Ugo Vaccaro, and Moti Yung. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology - Crypto'92*, volume 740 of *Lecture Notes in Computer Science*, pages 471 – 486, Santa Barbara, USA, August 1992. Springer - Verlag.
- [5] 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.
- [6] Yacine Challal, Abdelmajid Bouabdallah, and Hatem Bettahar. H₂A: Hybrid hash-chaining scheme for adaptive multicast source authentication of media-streaming. *Computer & Security*, 24(1):57 – 68, February 2005.
- [7] Seonho Choi. Denial of service resistant multicast authentication protocol with prediction hashing and one-way key chain. In *ISM 2005*, pages 701 – 706, Irvine, USA, December 2005. IEEE Press.

- [8] Scott Contini, Arjen K. Lenstra, and Ron Steinfeld. VSH: an efficient and provable collision resistant hash function. In *Advances in Cryptology - Eurocrypt'06*, volume 4004 of *Lecture Notes in Computer Science*, pages 165 – 182, Saint Petersburg, Russia, May 2006. Springer - Verlag.
- [9] Amir F. Dana, Radhika Gowaikar, Ravi Palanki, Babak Hassibi, and Michelle Effros. Capacity of wireless erasure networks. *IEEE Transactions on Information Theory*, 52(3):789 – 804, March 2006.
- [10] Yvo Desmedt, Yair Frankel, and Moti Yung. Multi-receiver/multi-sender network security: Efficient authenticated multicast/feedback. In *INFOCOM '92*, volume 3, pages 2045 – 2054, Florence, Italy, May 1992. IEEE Press.
- [11] Roberto Di Pietro, Stefano Chessa, and Piero Maestrini. Computation memory and bandwidth efficient distillation codes to mitigate DoS in multicast. In *SecureComm 2005*, pages 13 – 22, Athens, Greece, September 2005. IEEE Press.
- [12] James Chuan Fu and W. Y. Wendy Lou. *Distribution Theory of Runs and Patterns and its Applications*. World Scientific Publishing, 2003.
- [13] Rosario Gennaro and Pankaj Rohatgi. How to sign digital streams. In *Advances in Cryptology - Crypto'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 180–197, Santa Barbara, USA, August 1997. Springer-Verlag.
- [14] Phillippe Golle and Nagendra Modadugu. Authenticating streamed data in the presence of random packet loss. In *Network and Distributed Systems Security Symposium on*, pages 13 – 22, San Diego, USA, February 2001. Internet Society.
- [15] Venkatesan Guruswami. *List Decoding of Error-Correcting Codes*. Springer-Verlag, 2004.
- [16] 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.
- [17] Goce Jakimoski. *Primitives and Schemes for Non-Atomic Information Authentication*. PhD thesis, The Florida State University College of Arts and Sciences, Spring Semester 2006.
- [18] Chris Karlof, Naveen Sastry, Yaping Li, Adrian Perrig, and J. D. Tygar. Distillation codes and applications to DoS resistant multicast authentication. In *11th Network and Distributed Systems Security Symposium (NDSS)*, San Diego, USA, February 2004.
- [19] Jérôme Lacan and Jérôme Fimes. Systematic MDS erasure codes based on Vandermonde matrices. *IEEE Communications Letters*, 8(9):570 – 572, September 2004.
- [20] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and their Applications - Revised Edition*. Cambridge University Press, 2000.
- [21] Donggang Liu and Peng Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *NDSS'03*, pages 263 – 276, San Diego, USA, February 2003. The Internet Society.
- [22] Donggang Liu and Peng Ning. Multi-level μ TESLA: Broadcast authentication for distributed sensor networks. *ACM Transactions in Embedded Computing Systems*, 3(4):800 – 836, November 2004.
- [23] Donggang Liu, Peng Ning, Sencun Zhu, and Sushil Jajodia. Practical broadcast authentication in sensor networks. In *MobiQuitous 2005*, pages 118 – 129, San Diego, USA, July 2005. IEEE Press.
- [24] 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.
- [25] Florence Jessiem MacWilliams and Neil James Alexander Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.
- [26] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
- [27] 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.
- [28] 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.
- [29] 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.

- [30] 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.
- [31] 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.
- [32] 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.
- [33] Vern Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277 – 292, June 1999.
- [34] 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.
- [35] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. SPINS: Security protocols for sensor networks. *Wireless Networks*, 8(5):521 – 534, September 2002.
- [36] Adrian Perrig and J. D. Tygar. *Secure Broadcast Communication in Wired and Wireless Networks*. Kluwer Academic Publishers, 2003.
- [37] Josef Pieprzyk, Thomas Hardjono, and Jennifer Seberry. *Fundamentals of Computer Security*. Springer, 2003.
- [38] 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.
- [39] Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of Society for Industrial and Applied Mathematics*, 8(2):300 – 304, June 1960.
- [40] Pankaj Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *6th ACM Conference on Computer and Communications Security*, pages 93 – 100, Singapore, November 1999. ACM Press.
- [41] Rei Safavi-Naini and Huaxiong Wang. New results on multi-receiver authentication code. In *Advances in Cryptology - Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 527 – 541, Espoo, Finland, June 1998. Springer - Verlag.
- [42] Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In *Advances in Cryptology - Crypto'01*, volume 2139 of *Lecture Notes in Computer Science*, pages 355 – 367, Santa Barbara, USA, August 2001. Springer - Verlag.
- [43] Douglas R. Stinson. *Cryptography: Theory and Practice (Third Edition)*. Chapman & Hall/CRC, 2006.
- [44] Christophe Tartary and Huaxiong Wang. Efficient multicast stream authentication for the fully adversarial network. *International Journal of Security and Network (Special Issue on Cryptography in Networks)*, to appear. Inderscience.
- [45] Christophe Tartary and Huaxiong Wang. Efficient multicast stream authentication for the fully adversarial network. In *6th International Workshop on Information Security Applications*, volume 3786 of *Lecture Notes in Computer Science*, pages 108 – 125, Jeju Island, Korea, August 2005. Springer - Verlag.
- [46] 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.
- [47] C. K. Wong and Agnes Chan. Immediate data authentication for multicast resource constrained networks. In *10th Australasian Conference Information Security and Privacy*, volume 3574 of *Lecture Notes in Computer Science*, pages 113 – 121, Brisbane, Australia, July 2005. Springer - Verlag.
- [48] Chung Kei Wong and Simon S. Lam. Digital signatures for flows and multicasts. *IEEE/ACM Transactions on Networking*, 7(4):502 – 513, August 1999.
- [49] Maya Yajnik, Sue Moon, Jim Kurose, and Don Towsley. Measurement and modeling of the temporal dependence in packet loss. In *IEEE Conference on Computer Communications*, volume 1, pages 345 – 352, New York, USA, March 1999. IEEE Press.
- [50] Jean-Pierre Zanotti. Le code correcteur C.I.R.C. Available online at: <http://zanotti.univ-tln.fr/enseignement/divers/chapter3.html>.

A Design of Pads

In this appendix, we study the different pads to be used for our scheme. The field \mathbb{F}_{2^q} is used to represent polynomial coefficients while $\mathbb{F}_{2^{\tilde{q}}}$ is utilized for the MDS code. We denote \mathcal{P} the bit size of any packet $P_{i,j}$, \mathcal{H} the bit size of digests produced by h and s the bit size of signatures as in Sect. 3.

Polynomial Representation. These pads occur at Step 3.1 and Step 4 of Authenticator. At Step 3.1, the λ block hashes have to be padded. Each of them represents $n\mathcal{H}$ bits. At Step 4, the element $h_f\|\sigma$ is $\lambda\mathcal{H} + s$ bits long. In both cases, the padded value is to be represented by $\rho n + 1$ elements of \mathbb{F}_{2^q} . We obtain q as:

$$q = \left\lceil \frac{\max(n\mathcal{H}, \lambda\mathcal{H} + s)}{\rho n + 1} \right\rceil$$

Notice that, in practical applications, n will be much larger than λ . Therefore we will have: $n\mathcal{H} > \lambda\mathcal{H} + s$ and thus:

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

which proves that any element of \mathbb{F}_{2^q} is just a few hashes long.

Denote $b := \max(n\mathcal{H}, \lambda\mathcal{H} + s) \bmod \rho n + 1$. Thus the length of the pad at Step 3.1 is ℓ bits where:

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

The pad used at Step 4 then represents $(n - \lambda)\mathcal{H} - s + \ell$ bits.

Erasur Correcting Code. These pads occur at Step 1 and Step 2.2 of Authenticator. In both cases, we use the same finite field $\mathbb{F}_{2^{\tilde{q}}}$. Since the elements to be encoded at Step 2.2 are larger than those from Step 1, we first focus on the sooner.

Each message $(M_{i,1}, \dots, M_{i,\lceil \alpha n \rceil})$ is made of $\lceil \alpha n \rceil$ field elements and represents the concatenation $P'_{i,1} \|\dots\| P'_{i,n} \| 0^{\tilde{\ell}}$ (where $\tilde{\ell}$ is the length of the pad). By construction $P'_{i,1} \|\dots\| P'_{i,n}$ is $n(\mathcal{P} + \mathcal{H})$ bits long.

Denote $\tilde{b} := n(\mathcal{P} + \mathcal{H}) \bmod \lceil \alpha n \rceil$. Thus the length of the pad $\tilde{\ell}$ is:

$$\tilde{\ell} := \begin{cases} 0 & \text{if } \tilde{b} = 0 \\ \lceil \alpha n \rceil - \tilde{b} & \text{otherwise} \end{cases}$$

We get:

$$\tilde{q} = \left\lceil \frac{n(\mathcal{P} + \mathcal{H})}{\lceil \alpha n \rceil} \right\rceil \simeq \frac{\mathcal{P} + \mathcal{H}}{\alpha}$$

Concerning Step 1 of Authenticator, the quantity to be padded is $P_{\lambda,1} \|\dots\| P_{\lambda,n}$ which is $n\mathcal{P}$ bits long. In that case, the length of the pad is $n\mathcal{H} + \tilde{\ell}$ bits as we want to use the same MDS code (in particular the same field) for all coding operations.

Important Remark. The reader can notice that information stored in the public Table 1 is sufficient to compute of the previous pads.

B 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 $(\text{FID}_i, \lambda_i, n_i, \alpha_i, \beta_i, \rho_i, \mathcal{Q}_i(X), \text{DP}_i)$ where DP_i is the set of $\lambda_i n_i$ 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 4 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 $\text{FID}, \lambda, n, \alpha, \beta, \rho, \mathcal{Q}(X)$ and a set of received packets RP_{BID} (for some $\text{BID} \in \{1, \dots, \lambda\}$) such that:

- $\exists i / (\text{FID}, \lambda, n, \alpha, \beta, \rho, \mathcal{Q}(X)) = (\text{FID}_i, \lambda_i, n_i, \alpha_i, \beta_i, \rho_i, \mathcal{Q}_i(X))$.
Denote $\text{DP} = \{P_{1,1}, \dots, P_{\lambda,n}\} (= \text{DP}_i)$ the λn 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 4 of Authenticator.
- $|\text{RP}_{\text{BID}} \cap \text{AP}_{\text{BID}}| \geq \lceil \alpha n \rceil$ and $|\text{RP}_{\text{BID}}| \leq \lfloor \beta n \rfloor$.
- $\{P'_{\text{BID},1}, \dots, P'_{\text{BID},n}\} = \text{DynamicDecoder}(\text{PK}, \text{FID}, \text{BID}, n, \alpha, \beta, \rho, \mathcal{Q}(X), \text{KnownHashes}, \text{HashCodeword}, \text{RP}_{\text{BID}})$ where $P'_{\text{BID},j} \neq P_{\text{BID},j}$ for some $j \in \{1, \dots, n\}$.

In the previous statement, AP_{BID} denotes the subfamily of AP related to block number BID. We would also like to draw the reader's attention to the fact that \mathcal{O} does not have any influence either on the boolean value KnownHashes or the content of HashCodeword as these two elements are updated by the receiver personally. Thus, \mathcal{O} must succeed in both following cases:

- The value of KnownHashes is FALSE
- The value of KnownHashes is TRUE

Case A. Due to the design of DynamicDecoder, the family $\{P'_{\text{BID},1}, \dots, P'_{\text{BID},n}\}$ has been returned as an output of DecoderBlock. Since $|\text{RP}_{\text{BID}} \cap \text{AP}_{\text{BID}}| \geq \lceil \alpha n \rceil$ and $|\text{RP}_{\text{BID}}| \leq \lfloor \beta n \rfloor$, Step 1 ends successfully. Assume that the digital signature is unforgeable and the hash function is collision resistant. Applying Theorem 2 from [45], VerifySignatureFamily recovers the signature σ and stores in HashBlock the λ block hashes h_1, \dots, h_λ corresponding to DP.

For the same reason as before, the string $h(C_{\text{BID},1}) \parallel \dots \parallel h(C_{\text{BID},n})$ will be included in the list L returned by PolyReconstruct at Step 3. Since h is collision resistant, the values $h_{\text{BID}}^1, \dots, h_{\text{BID}}^n$ at the end of Step 4 contain the original digests, i.e.: $\forall j \in \{1, \dots, n\} h_{\text{BID}}^j = h(C_{\text{BID},j})$.

Because h is collision resistant and $|\text{RP}_{\text{BID}} \cap \text{AP}_{\text{BID}}| \geq \lceil \alpha n \rceil$, the element \mathcal{C}' has at least $\lceil \alpha n \rceil$ non-erased coordinates and each of them is consistent with the original codeword $(C_{\text{BID},1}, \dots, C_{\text{BID},n})$. Therefore, at the end of Step 6, the family $\{P'_{\text{BID},1}, \dots, P'_{\text{BID},n}\}$ is equal to the original family $\{P_{\text{BID},1}, \dots, P_{\text{BID},n}\}$. We get a contradiction with our hypothesis which stipulated:

$$\exists j \in \{1, \dots, n\} \quad P'_{\text{BID},j} \neq P_{\text{BID},j}$$

Therefore, either the hash function is not collision resistant or the digital signature is not secure.

Case B. The family $\{P'_{\text{BID},1}, \dots, P'_{\text{BID},n}\}$ has been returned after a query to FilterElements. Since the boolean KnownHashes is reinitialized to FALSE each time the first block of a new family is to be processed, we deduce that DynamicDecoder successfully queried DecoderBlock for family tag FID. Denote $\widetilde{\text{BID}}$ the last block tag such that DecoderBlock was successfully run. We have: $\widetilde{\text{BID}} = \text{BID} - 1$ for some $\mu \geq 1$.

For ease of explanation, we assume $\mu = 1$. That is, DecoderBlock was successfully queried by the receiver for block tag $\text{BID} - 1$. Thus the unforgeability of the digital signature involves that the receiver obtained the λ block hashes corresponding to DP while running VerifySignatureFamily and then the n original hashes $h(C_{\text{BID}-1,1}), \dots, h(C_{\text{BID}-1,n})$ at Step 4 (see Case A). Thus, the non-erased coordinates of \mathcal{C}' are consistent with $(C_{\text{BID}-1,1}, \dots, C_{\text{BID}-1,n})$ in at least $\lceil \alpha n \rceil$ positions. Therefore, the corrected message is the original one: $(M_{\text{BID}-1,1}, \dots, M_{\text{BID}-1, \lceil \alpha n \rceil})$. Thus, the elements memorized into HashCodeword at the end of the query on block $\text{BID} - 1$ are the original hashes $h(C_{\text{BID},1}), \dots, h(C_{\text{BID},n})$.

Now, we study the case of block BID which is under attack by \mathcal{O} . Assume that some elements of the output $\{P'_{\text{BID},1}, \dots, P'_{\text{BID},n}\}$ are empty. Due to the design of FilterElements, this happens if and only if the decoding process at Step 3.1 did not occur. That means the element \mathcal{C}' had less than $\lceil \alpha n \rceil$ non-erased coordinates at the end of Step 2.2. Nevertheless RP_{BID} contains at least $\lceil \alpha n \rceil$ elements of AP_{BID} and $\text{HashCodeword} = (h(C_{\text{BID},1}), \dots, h(C_{\text{BID},n}))$. Therefore \mathcal{C}' must have at least $\lceil \alpha n \rceil$ non-erased positions. We obtain a contradiction. Therefore, none of the packets $\{P'_{\text{BID},1}, \dots, P'_{\text{BID},n}\}$ is empty. As a consequence, they were obtained by removing the pad from some message $(M'_{\text{BID},1}, \dots, M'_{\text{BID}, \lceil \alpha n \rceil})$ at Step 3.2.

Since $P'_{\text{BID},j}$ is an incorrect non-empty packet we have $P'_{\text{BID},j} \neq P_{\text{BID},j}$. Thus the message $(M'_{\text{BID},1}, \dots, M'_{\text{BID},\lceil \alpha n \rceil})$ is different from the original message $(M_{\text{BID},1}, \dots, M_{\text{BID},\lceil \alpha n \rceil})$. It follows that at least one of the non-erased coordinates of the codeword $(C'_{\text{BID},1}, \dots, C'_{\text{BID},n})$ at Step 3 is different from the corresponding coordinate of $(C_{\text{BID},1}, \dots, C_{\text{BID},n})$. That is:

$$\exists \theta \in \{1, \dots, n\} / \begin{cases} \text{FID} \parallel \text{BID} \parallel \theta \parallel C'_{\text{BID},\theta} \parallel \mathcal{B}_\theta \parallel \mathcal{F}_\theta \in \text{RP}_{\text{BID}} \\ C'_{\text{BID},\theta} \neq C_{\text{BID},\theta} \end{cases} \quad \text{for some } \mathcal{B}_\theta \text{ and } \mathcal{F}_\theta$$

Nevertheless, $C'_{\text{BID},\theta}$ has been selected at Step 2.2. Therefore: $h(C'_{\text{BID},\theta}) = \text{HashCodeword}(\theta)$. Nonetheless, we previously showed that HashCodeword contained the original digests of block BID. In particular: $\text{HashCodeword}(\theta) = h(C_{\text{BID},\theta})$. So: $h(C'_{\text{BID},\theta}) = h(C_{\text{BID},\theta})$ while $C'_{\text{BID},\theta} \neq C_{\text{BID},\theta}$. This contradicts the collision resistance of h .

Now, we would like to argue about the case $\widetilde{\text{BID}} = \text{BID} - \mu$ when $\mu > 1$. This means that DynamicDecoder only queried (successfully) FilterElements for blocks $\text{BID} - \mu + 1, \dots, \text{BID} - 1$. In this case, it is easy to see, by recursion on the block tag value, that the different updates of HashCodeword are consistent with the original block hashes of DP. That is, at the end of query on block tag $\widetilde{\text{BID}}$, HashCodeword contains the digests for block $\widetilde{\text{BID}} + 1$ (see before); at the end of query on block tag $\widetilde{\text{BID}} + 1$, HashCodeword contains the digests for block $\widetilde{\text{BID}} + 2$ and so on. Thus at the beginning of the query on the block tag BID attacked by \mathcal{O} , HashCodeword contains the original hashes $h(C_{\text{BID},1}), \dots, h(C_{\text{BID},n})$. Therefore we obtain a collision for the hash function as previously.

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} will succeed if one of the following holds:

- I. Authenticator was never queried on input $\text{FID}, \lambda, n, \alpha, \beta, \rho, \mathcal{Q}(X)$ and the decoding algorithm DynamicDecoder does not reject RP_{BID} , i.e. $\{P'_{\text{BID},1}, \dots, P'_{\text{BID},n}\} \neq \emptyset$ where $\{P'_{\text{BID},1}, \dots, P'_{\text{BID},n}\} = \text{DynamicDecoder}(\text{PK}, \text{FID}, \text{BID}, n, \alpha, \beta, \rho, \mathcal{Q}(X), \text{KnownHashes}, \text{HashCodeword}, \text{RP}_{\text{BID}})$.
- II. Authenticator was queried on input $\text{FID}, \lambda, n, \alpha, \beta, \rho, \mathcal{Q}(X)$ for some data packets $\text{DP} = \{P_{1,1}, \dots, P_{\lambda,n}\}$. Nevertheless, the output of DynamicDecoder verifies $P'_{\text{BID},j} \neq P_{\text{BID},j}$ for some $\text{BID} \in \{1, \dots, \lambda\}$ and some $j \in \{1, \dots, n\}$.

Case I. As in Point (1), we have two cases to consider according to the value of KnownHashes.

• Sub-case IA. If the value of KnownHashes is FALSE then DynamicDecoder output $\{P'_{\text{BID},1}, \dots, P'_{\text{BID},n}\}$ after querying DecoderBlock. Since Step 2 of the latter algorithm had to terminate successfully, VerifySignatureFamily output a pair $(h(\text{FID} \parallel \lambda \parallel n \parallel \alpha \parallel \beta \parallel \rho \parallel h_1 \parallel \dots \parallel h_\lambda), \sigma)$ such that:

$$\text{Verify}_{\text{PK}}(h(\text{FID} \parallel \lambda \parallel n \parallel \alpha \parallel \beta \parallel \rho \parallel h_1 \parallel \dots \parallel h_\lambda), \sigma) = \text{TRUE}$$

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

If \mathcal{O} queried Authenticator for family tag FID then denote $(\text{FID}, \tilde{\lambda}, \tilde{n}, \tilde{\alpha}, \tilde{\beta}, \tilde{\rho}, \tilde{\mathcal{Q}}(X))$ his query. By hypothesis, we have:

$$(\text{FID}, \tilde{\lambda}, \tilde{n}, \tilde{\alpha}, \tilde{\beta}, \tilde{\rho}, \tilde{\mathcal{Q}}(X)) \neq (\text{FID}, \lambda, n, \alpha, \beta, \rho, \mathcal{Q}(X))$$

Without loss of generality, we can assume that once the parameters $\lambda, n, \alpha, \beta, \rho$ are known, the polynomial $\mathcal{Q}(X)$ used to represent field elements is uniquely determined. So, we obtain:

$$(\tilde{\lambda}, \tilde{n}, \tilde{\alpha}, \tilde{\beta}, \tilde{\rho}) \neq (\lambda, n, \alpha, \beta, \rho)$$

Therefore, the pair output by VerifySignatureFamily is a forgery of the signature scheme.

• Sub-case IB. If the value of KnownHashes is TRUE then it means that there exists an earlier block (numbered $\widetilde{\text{BID}} < \text{BID}$) which was used to successfully verify a signature for parameters $\text{FID}, \lambda, n, \alpha, \beta, \rho, \mathcal{Q}(X)$. In this case, we can perform on that earlier block the same reasoning as in the previous case to obtain a contradiction with the security of either the signature scheme or the hash function.

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

C Proof of Theorem 2

We first show that our scheme allows recovery of all data packets as part of this demonstration will be used to deduce the bound on the number of signature verifications to be performed for the whole family.

Packet Recovery. Let FID be fixed. Since (α, β) is accurate, for each BID, the receiver obtains, as set of received packets $\overline{\text{RP}}_{\text{BID}}$, at least $\lceil \alpha n \rceil$ original elements amongst a total not exceeding $\lfloor \beta n \rfloor$ pieces of data.

- We start focusing on the first block (i.e. $\text{BID} = 1$). As it is the first block of a new family, DynamicDecoder queries DecoderBlock as a subroutine. Since RP_1 has at least $\lceil \alpha n \rceil$ correct elements and $|\text{RP}_1| \leq \lfloor \beta n \rfloor$, Step 1 ends successfully. Applying Theorem 2 of [45], VerifySignatureFamily recovers the family signature and fills in HashBlock with the digests of the original blocks, i.e.:

$$\forall i \in \{1, \dots, \lambda\} \text{HashBlock}(i) = h(h(C_{i,1}) \parallel \dots \parallel h(C_{i,n}))$$

The accuracy of (α, β) involves that the list L output by Poly-Reconstruct at Step 3 includes the string $h(C_{i,1}) \parallel \dots \parallel h(C_{i,n})$. Due to the collision resistance of the hash function and the content of HashBlock, we deduce that h_1^1, \dots, h_1^n from Step 4 are such as: $\forall i \in \{1, \dots, n\} h_1^i = h(C_{1,i})$.

The collision resistance of h also implies that the non-empty coordinates of \mathcal{C}' (at the end of Step 5) are consistent with those of the original codeword $(C_{1,1}, \dots, C_{1,n})$. The accuracy of (α, β) involves that there are at least $\lceil \alpha n \rceil$ non-erased elements. Thus, the MDS code can correct the erasures at Step 6 so that the message recovered by the receiver is the original one $(M_{1,1}, \dots, M_{1,n})$.

As a consequence, the output $\{P'_{1,1}, \dots, P'_{1,n}\}$ of DecoderBlock is the original set $\{P_{1,1}, \dots, P_{1,n}\}$ (total recovery for block 1), the boolean KnownHashes is set to TRUE and HashCodeword is such as: $\forall i \in \{1, \dots, n\} \text{HashCodeword}(i) = h(C_{2,i})$.

- We consider the second block (i.e. $\text{BID} = 2$). Since KnownHashes is TRUE, DynamicDecoder queries FilterElements as a subroutine. The non-empty coordinates of \mathcal{C}' at the end of Step 2.2 must be consistent with the content of HashCodeword $= (h(C_{2,1}), \dots, h(C_{2,n}))$. Since (α, β) is accurate, RP_2 contains at least $\lceil \alpha n \rceil$ original elements. Thus, the MDS code can correct the erasures at Step 3.1 so that the message recovered by the receiver is the original one $(M_{2,1}, \dots, M_{2,n})$.

As a consequence, the output $\{P'_{2,1}, \dots, P'_{2,n}\}$ of DecoderBlock is the original set $\{P_{2,1}, \dots, P_{2,n}\}$ (total recovery for block 2), the boolean KnownHashes is still set to TRUE and HashCodeword is such as: $\forall i \in \{1, \dots, n\} \text{HashCodeword}(i) = h(C_{3,i})$.

- We see that when iterating this process we obtain total recovery of data for blocks $3, \dots, \lambda$ as well.

Signature Verification. The previous proof showed that VerifySignatureFamily is only queried once per family of λ blocks. Therefore the number of signature verifications is upper bounded by the size of the list constructed by VerifySignatureFamily. We apply Theorem 3 from [46] to get:

1. the list size is upper bounded by $U(n)$.
2. $U(n) \in O(1)$ as a function of the block length n .

Since $U(n)$ is independent from the family length λ , we deduce that $U(n) \in O(1)$ as a function of λ as well.