Authentication of Digital Streams

Christophe Tartary, *Member, IEEE*, Huaxiong Wang, and San Ling

***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. Recently, many coding theory based protocols have been developed to treat the stream authentication problem over such a channel. In this paper, our goal is to provide a general coding approach for multicast stream authentication. We design an authentication protocol which combines *any* list recoverable code (provided some conditions on its construction parameters). We demonstrate that the previous schemes can be viewed as instances of our construction when a Reed–Solomon code is used as a list recoverable code. In such settings, we also show that our approach leads to a better upper bound on the number of signature verification queries for *each* receiver.**

***Index Terms*—Adversarial network, list recoverable codes, polynomial reconstruction problem, stream authentication.**

I. Introduction

WITH the expansion of communication networks, broadcasting has become a major way of distributing digital content to a large audience over public communication channels such as the Internet. Video-conferences, air traffic control, software updates and stock quotes are examples of applications based on broadcast. Unfortunately, large-scale broadcasts prevent lost content from being retransmitted for two reasons. First, the size of the communication group involves that a single deletion could lead to an overwhelming number of redistribution requests at the sender end. Second, a feedback channel for those requests may not even exist as in the case of satellite television. In addition to these drawbacks, the network can be under the influence of malicious users altering the data stream. Thus, the security of broadcast transmission schemes depends on both the network properties and the opponents' computational power. In this paper, we will consider scenarios where opponents have bounded computational abilities.

Some applications, e.g., television and stock market, will diffuse a long stream of data whose content is to be authenticated by the receivers within a short period of time upon reception. Since many multicast protocols transfer private or sensitive information, nonrepudiation of the sender is required. In addition, most channels used for multicast only provide a best effort delivery of data like the Internet with the User Datagram Protocol [1].

Two important concerns of the multicast authentication problem are the network bandwidth availability and the receivers' computational abilities. Indeed, large packets may create irregular throughput of data which sometimes results in congestion of the network information flow. On the other hand, receivers with limited computational abilities will require more time to authenticate data delaying the stream play. Therefore, a stream authentication protocol should aim to minimize both packet[1] overhead and computational cost.

The stream authentication problem has been extensively studied (see, for example, the survey [2]). The simplest technique to ensure nonrepudiation of information is to use a signature to sign each data packet (also known as the sign-each approach). However, signature schemes are time expensive to generate and verify which makes this idea impractical for stream authentication. As a consequence, a common approach is to generate a single signature and to amortize its communication and computation overheads over several packets using hash chains for instance.

By appending the digest of each packet to several followers according to some specific patterns, Perrig *et al.* [3], [4], Golle and Modadugu [5] and Miner and Staddon [6] designed schemes dealing with packet loss. One signature was generated from time to time to ensure nonrepudiation of data. In these papers, the network packet loss behavior was modeled by a k-state Markov chain [7], [8] which provided bounds on the packet authentication probability. Unfortunately, all these schemes rely on the reception of signature packets.

To overcome this problem, one solution is to split the signature into k smaller parts where only ℓ of them $(\ell < k)$ are sufficient for recovery. Along this line, several schemes were developed [9]–[13] but none of them tolerates a single packet injection. In 2003, Lysyanskaya *et al.* [14][2] designed a technique resistant to packet loss and data injections using Reed–Solomon (RS) codes [16] where the number of signature verifications to be performed per block[3] turns out to be $O(1)$ as a function of the block length n.

Manuscript received August 29, 2009; revised April 20, 2011; accepted May 22, 2011. Date of current version August 31, 2011. C. Tartary was supported in part by the National Natural Science Foundation of China under Grants 61033001, 61061130540, 61073174, and 61050110147 (International Young Scientists program); in part by the National Basic Research Program of China under Grants 2007CB807900 and 2007CB807901; in part by the Danish National Research Foundation; and in part by the National Natural Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation (CTIC) within which part of this work was performed. H. Wang and S. Ling were supported by the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

C. Tartary is with the Institute for Theoretical Computer Science within the Institute for Interdisciplinary Information Sciences at Tsinghua University, Beijing, 100084, China (e-mail: ctartary@mail.tsinghua.edu.cn).

H. Wang and S. Ling are with the Division of Mathematical Sciences in the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore (e-mail: hxwang@ntu.edu.sg; lingsan@ntu.edu.sg).

Communicated by T. Fujiwara, Associate Editor for Complexity and Cryptography.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2011.2161960

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

[2]An updated version of this work recently appeared in [15].

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

This approach was later extended in [17]–[21]. In 2004, Karlof *et al.* developed a protocol called PRABS [22] using a Maximum Distance Separable (MDS) code along with a one-way accumulator [23] based on a Merkle hash tree [24] requiring less signature verifications than the techniques from [14], [17]–[21]. Unfortunately, PRABS's augmented packets[4] must carry $\lceil \log_2 n \rceil$ hash values which is much larger than for those constructions. It should be noticed that this drawback comes from the use of the tree which means that techniques from [25]–[28] also have this problem. The reader can find a more detailed survey of authentication protocols in [29].

In this paper, we propose a new coding approach to the stream authentication problem. Our technique relies on list recoverable codes and it is based on the following observation. When the adversary pollutes the communication channel, the receiver gets several candidate values for some packets. Since performing exhaustive search on these elements is computationally prohibitive, one seeks a way to reduce the number of possible packets to a small number. An (ℓ, L)-list recoverable code is an error-correcting code such that if there are at most ℓ values per codeword coordinate then there are at most L codewords consistent with these elements (a more rigorous definition can be found in Section III-B-II). Note that the adversary may inject elements in such a way that some packets may have more than ℓ candidates. In such a situation, we will delete all the corresponding coordinate candidates and consider this position as an erasure.

The benefits of our approach are twofold. First, it provides with a black-box construction for stream authentication protocols based on list recoverable codes. That is, any list recoverable code (with suitable parameters ℓ and L) can be utilized for our scheme while preserving its security. Second, we will demonstrate that the constructions from [14], [17]–[21] can be treated as particular cases of our approach where the deletion process does not occur. In fact, we will even show that this deletion process, when performed on the RS code used in [14], [17]–[21], leads to a smaller signature verification cost than in those papers. This result shows that our technique outperforms those constructions. In addition, we provide explicit values for a critical parameter of the RS decoding algorithm in the context of broadcasting. Such a study has never been done before [14], [17]–[21] since the decoding algorithm was rather regarded as a black-box. These values provide essential information to efficiently implement our scheme using RS codes.

This paper is organized as follows. In Section II, we recall our previous studies on broadcast authentication and present the reader with the new approach undertaken in this paper. In Section III, we introduce the mathematical tools needed in this paper. In Section IV, we describe our coding-based protocol for the stream authentication problem. Its security and efficiency are studied in Section V. In Section VI, we compare the benefits with respect to the RS code-based constructions from [14], [17]–[21] and we refine the parameters of the RS decoding algorithm for practical implementations. The last section will summarize our contributions to the broadcast authentication problem.

[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.

II. Related Work and Paper Contribution

The aim of a stream authentication protocol is to enable the data receiver to authenticate the origin of information. The data is distributed through a network where some nodes may be under control of an adversary whose goal can be either to prevent some participants from validating genuine packets or to have some receivers identify incorrect data as authentic. In this context, several parameters must be taken into account: packet overhead, computational power and memory capacity of the participants, type of streaming (live or delayed).

In [18] and [19], we presented protocols allowing the receiver to perform authentication of information and reconstruction of the whole data stream. Our goal was to study the trade-off between computational costs and reconstruction of information. The purpose of [21] was to focus on the trade-off between packet overhead and number of signature verification queries. Those constructions dealt with live distribution of data. However, multicast can be used to broadcast delayed streams. One can assume that the sender knows a longer part of the stream than during live distribution. The work of [17], [20], [30] assumed that the sender could buffer $\lambda(\geq 1)$ blocks at a time to reduce the number of signature verification queries and the packet overhead even further.

In all those papers [17]–[21], [30], we used the Guruswami-Sudan algorithm to deal with injections of bogus data by the adversary as in [14]. As said above, our investigations were focused on finding different practical trade-offs between packet overhead, signature verification complexity, computational cost and memory requirements. In the current paper, our study is to focus on another central—though more theoretical in its treatment—issue for authentication: dealing with adversarial injections of data. This is crucial as any protocol dealing with this problem can be used as a subroutine of any existing authentication scheme. Apart from the sign-each approach which is computationally prohibitive, there currently exist only two techniques in the literature which can handle those injections: cryptographic accumulators and the Guruswami-Sudan algorithm. As explained in Section I, the design of current accumulators does not satisfy the requirements of multicast authentication: those based on Merkle hash trees have too large an overhead while a construction like Nguyen's [31] requires the use of pairings on elliptic curves [32] which is too slow [33] for quick authentication.

The starting point of this work was the study of the original goal of the Guruswami-Sudan algorithm: performing list-decoding of RS codes. In this paper, we first show that we can design similar secure protocols for any list recoverable protocol (provided some conditions on their parameters). That is, the security of the construction only relies on the list recoverable property of the code and the existence of an efficient algorithm to construct that list. This has the advantage to provide a black-box design for such authentication protocols.

The second point we present in this paper is the column deletion process. When analyzing the use of the Guruswami-Sudan algorithm in the multicast context, one notices that the receiver inputs the whole set of collected data. Our approach is based on the observation that a packet has a single correct value. Therefore, if the receiver collects a lot of candidates for a particular packet, then the noise is important for that element. Thus, it may be computationally less expensive to treat the packet as an erasure rather than inputting the whole set of candidate values to the list decoding algorithm. However, these potential extra erasures lead to other conditions for the code parameters. In Sections IV and V, we treat the problems of black-box list recoverable codes and column deletions at the same time.

We illustrate our approach using RS codes in Section VI. We first deduce a lower bound on the column height to perform deletions. This value, valid for all receivers, is based on the network parameters. Second, we demonstrate that this value turns the RS code into a list recoverable code where the receivers can use the Guruswami-Sudan algorithm with identical parameters. Third, we deduce an upper bound on the size of the list output by that reconstruction algorithm and we show that this deletion-based bound is tighter than the value computed in [18] which was also valid for our different constructions quoted above as well as the original scheme from [14] (Sections VI-A and VI-B). This result justifies our original idea of ignoring very noisy packets by removing them from the set of elements input to the list recovery algorithm.

When studying the survey done by McEliece on the Guruswami-Sudan algorithm [34], one notices that there is an important parameter called the interpolation multiplicity which needs to be specified. However, no study of this parameter seems to have ever been done in the context of multicast authentication. We give a brief survey in Section VI-C-1 and we show the explicit values to be chosen to run this reconstruction algorithm efficiently. Remark that the result of the analysis can also be used with any construction running the Guruswami-Sudan algorithm as a subroutine. In Section VI-C-2, we use those critical values to deduce a second upper bound on the list size output by the algorithm and we show, on an example, that this new bound is tighter than in [18] as well. In this work, we also demonstrate that the packet overhead of our construction using column deletions is the same as for [14], [18].

We can summarize our contribution as follows. First, it generalizes the constructions from [14], [17]–[21] to the family of list recoverable codes and it introduces the idea of performing deletion of columns where the number of candidate values is large (i.e., we remove very noisy positions before reconstructing the list of consistent codewords/signature verification candidates). We get a black-box construction for stream authentication where any list recoverable code (with suitable parameters) can be used while preserving correctness and security of the scheme. Second, it shows that, when using RS codes, this deletion process turns the code into a list recoverable code having smaller list size. Third, the new bounds on the list size we obtain are tighter than existing ones. Fourth, this work provides explicit choices of the interpolation parameter (which is an essential parameter to run the Guruswami-Sudan algorithm) in the context of multicast authentication.

III. Mathematical Background

In this section, we review the tools that we use to ensure the security and to analyze the efficiency of our construction (see Section V). We also give an overview of the Guruswami-Sudan algorithm [35] that is used in [14], [17]–[21].

A. Cryptographic Primitives

Before presenting the primitives to be used in our protocol, we need to recall the following definition:

Definition III.1 ([36]): A function $f : \mathbb{N} \rightarrow \mathbb{R}^+$ is said to be negligible if for every positive polynomial $p(\cdot)$ there exists an integer N such that for all $n > N$, we have

$$f(n) < \frac{1}{p(n)}.$$

The nonrepudiation of the sender will be provided using digital signatures while the integrity of the data stream will be checked by applying hash functions.

1) Digital Signatures:

Definition III.2 ([37]): A signature scheme (or digital signature) is a five-tuple of sets $(\mathfrak{P}, \mathfrak{A}, \mathfrak{K}, \mathfrak{S}, \mathfrak{V})$ where the following conditions are satisfied:

1) $\mathfrak{P}$ is a finite set of possible messages.
2) $\mathfrak{A}$ is a finite set of possible signatures.
3) $\mathfrak{K}$, the keyspace, is a finite set of possible keys.
4) For each (SK,PK) $\in \mathfrak{K}$, there is a signing algorithm $\mathrm{Sign}_{\mathrm{SK}} \in \mathfrak{S}$ and a corresponding verification algorithm $\mathrm{Verify}_{\mathrm{PK}} \in \mathfrak{V}$. Each $\mathrm{Sign}_{\mathrm{SK}} : \mathfrak{P} \rightarrow \mathfrak{A}$ and $\mathrm{Verify}_{\mathrm{PK}} : \mathfrak{P} \times \mathfrak{A} \rightarrow \{\mathrm{TRUE}, \mathrm{FALSE}\}$ are functions such that the following equation is satisfied for every message $x \in \mathfrak{P}$ and for every signature $y \in \mathfrak{A}$
$$\mathrm{Verify}_{\mathrm{PK}}(x, y) = \begin{cases} \mathrm{TRUE} & \text{if } y = \mathrm{Sign}_{\mathrm{SK}}(x) \\ \mathrm{FALSE} & \text{if } y \neq \mathrm{Sign}_{\mathrm{SK}}(x) \end{cases}.$$
A pair (x, y) with $x \in \mathfrak{P}$ and $y \in \mathfrak{A}$ is called a signed message.

Note that, for every (SK,PK) $\in \mathfrak{K}$, $\mathrm{Sign}_{\mathrm{SK}}$ and $\mathrm{Verify}_{\mathrm{PK}}$ should be polynomial-time functions. $\mathrm{Verify}_{\mathrm{PK}}$ will be a public function while $\mathrm{Sign}_{\mathrm{SK}}$ will remain private. For concision in the remaining of this paper, we denote a signature scheme as a triple (KeyGen $\mathrm{Sign}_{\mathrm{SK}}$, $\mathrm{Verify}_{\mathrm{PK}}$) where KeyGen is the key generation algorithm taking as input the security parameter $\mathcal{S}$ representing the bit length of the signature and outputting a pair of elements (SK,PK) such that SK is the signer secret key while PK is the public key to be used by the verifier.

Definition III.3 ([37]): A digital signature (KeyGen, $\mathrm{Sign}_{\mathrm{SK}}, \mathrm{Verify}_{\mathrm{PK}}$) is said to be secure against chosen message attack if no Probabilistic Polynomial-Time (PPT) opponent $\mathcal{O}$ can win with non-negligible probability (as a function of the signature length $\mathcal{S}$) the following game:

1) $\mathcal{O}$ is given an oracle simulating $\mathrm{Sign}_{\mathrm{SK}}$ (but $\mathcal{O}$ does not have access to SK).
2) $\mathcal{O}$ chooses a polynomial number of messages $m_1, \ldots, m_\xi$ (ξ being a polynomial in $\mathcal{S}$) and queries the signing oracle to obtain their signatures. The messages are constructed adaptively, i.e., $\mathcal{O}$ chooses m_{i+1} after receiving the signature on m_i.
3) $\mathcal{O}$ constructs a pair (m, σ) such that: $\forall i \in \{1, \ldots, \xi\}\ m \neq m_i$.

The opponent $\mathcal{O}$ wins if: $\mathrm{Verify}_{\mathrm{PK}}(m, \sigma) = \mathrm{TRUE}$.

In most situations, security against known message attacks (i.e., $\mathcal{O}$ does *not* choose the messages to be signed by the oracle) is sufficient as $\mathcal{O}$ is only a channel eavesdropper. Nevertheless, in some cases, $\mathcal{O}$ has the ability to choose those messages. Consider the case of two TV stations emitting programs through the same satellite TV provider. Each station may try to use its own data to attack its rival. Thus, in our constructions, we will assume that our digital signature is secure according to Definition III.3.

2) Hash Functions:

Definition III.4 ([38]): A hash function is a function h which has, as a minimum, the following two properties:

1) Compression: h maps an input m of arbitrary finite bit length, to an output $h(m)$ of fixed bit length $\mathcal{H}$.
2) Ease of computation: Given h and an input m, $h(m)$ can be computed in polynomial-time as a function of $\mathcal{H}$.

The output $h(m)$ is called hash or digest of the message m.

We now present the property that the hash functions for our scheme will be assumed to have.

Definition III.5 ([38]): A hash function h as defined above is said to be collision resistant if no PPT opponent $\mathcal{O}$ can construct two different messages m_1 and m_2 such that $h(m_1) = h(m_2)$ with non-negligible probability as a function of $\mathcal{H}$.

B. Coding Theory

1) Maximum Distance Separable Codes: Since packets may be dropped during transmission, using an erasure correcting code will enable recovery of missing elements as in [18], [20], [21]. In our construction, we use linear codes.

A linear code of length N, dimension K and minimum distance D is denoted by $[N, K, D]$. The Singleton bound states that any $[N, K, D]$ code satisfies: $D - 1 \leq N - K$ [16]. It is known that any $[N, K, D]$ code can correct up to $D - 1$ erasures [39]. Thus, an $[N, K, D]$ 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 [16]. Even though our protocol works with any family of MDS codes, we suggest to use the codes created by Lacan and Fimes [40] for their efficient encoding and decoding as explained in [18].

2) List Recoverable Codes: In order to deal with injections of bogus elements into the network, we use a list recoverable code. These codes were first introduced by Elias [41] and Wozencraft [42] to treat the following problem. If we have more than 50% of errors in a codeword coordinates, then unique decoding is impossible. Nonetheless, in this situation, one would like to obtain a list of consistent messages.

Definition III.6 ([43]): For $\lambda \in [0, 1)$ and integers $L \geq \ell \geq 1$, a code $\mathcal{C}$ of length n over an alphabet Σ is said to be (λ, ℓ, L)-list recoverable if for all lists $\mathcal{L}_1, \ldots, \mathcal{L}_n$ of elements of Σ having size at most ℓ, there are at most L codewords $(C_1, \ldots, C_n) \in \mathcal{C}$ such that $C_i \in \mathcal{L}_i$ for at least $\lambda\, n$ indices i.

C. Polynomial Reconstruction Problems

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

Polynomial Reconstruction Problem:

Input: Integers D, T and N points $\{(x_i, y_i)\}_{i \in \{1,\ldots,N\}}$ where $x_i, y_i \in F$ for a field F.

Output: All univariate polynomials $P(X) \in F[X]$ of degree at most D such that $y_i = P(x_i)$ for at least T values of $i \in \{1, \ldots, N\}$.

In 1999, Guruswami and Sudan developed an algorithm called Poly-Reconstruct to solve the PRP [35]. Poly-Reconstruct has an adjustable integer parameter m called the *interpolation multiplicity*. This algorithm works in two steps:

1) The interpolation step. The decoder constructs a bivariate polynomial $Q(X, Y)$ with the property that $Q(X, Y)$ has a zero of multiplicity m at each of the N points and for which the $(1, D - 1)$ weighted degree is as small as possible.
2) The factorization step. The decoder finds all factors of $Q(X, Y)$ of the form $Y - P(X)$, where $P(X)$ is a polynomial of degree at most $D - 1$ and returns the list of all such polynomials $P(X)$.

This algorithm exhibits the following characteristics:

Theorem III.7 ([44]): The algorithm Poly-Reconstruct can solve the PRP in polynomial-time in N for any field F provided: $T > \sqrt{D\,N}$. In addition, the size of the output list is upper bounded by $\lfloor \ell / D \rfloor$ where

$$r := 1 + \left\lfloor \frac{D\,N + \sqrt{D^2\,N^2 + 4\,(T^2 - D\,N)}}{2\,(T^2 - D\,N)} \right\rfloor$$
$$\ell := r\,T - 1.$$

Guruswami demonstrated that Poly-Reconstruct runs in time quadratic in N while the size of the list is at most quadratic in N (see Theorem 6.12 from [44]). When also taking the interpolation multiplicity into account, its running time is $O(N^2\,m^4)$ [34]. In particular, as a function of the interpolation multiplicity (i.e., when N is regarded as a constant), its running time is $O(m^4)$. Algorithms for implementing Poly-Reconstruct and improvements can be found in [45], [46].

Remark III.8: In coding theory, one usually assumes that the x_i's entered as input of the PRP are pairwise distinct (see Section VI-A for an example). This condition forms a weaker variant of the PRP called the noisy PRP [47]. Thus, decoding is reduced to solving an instance of the noisy PRP. As trivially observed in [47], PolyReconstruct can be used in that situation as well.

IV. Design of the Stream Authentication Problem

A. Network Model

We assume that the communication channel is under control of an opponent $\mathcal{O}$ who can drop and rearrange packets. He is also allowed to inject bogus data into the network. This model is called the unsecured communication channel [38]. Note that it also corresponds to the Dolev-Yao threat model [48].

Since our primary concern is the multicast authentication problem, we can assume that a reasonable number of original augmented packets reach the receivers and not too many incorrect elements are injected by $\mathcal{O}$. Indeed, if too many original packets are dropped then strengthening data transmission is the main issue as it is likely that the few packets successfully

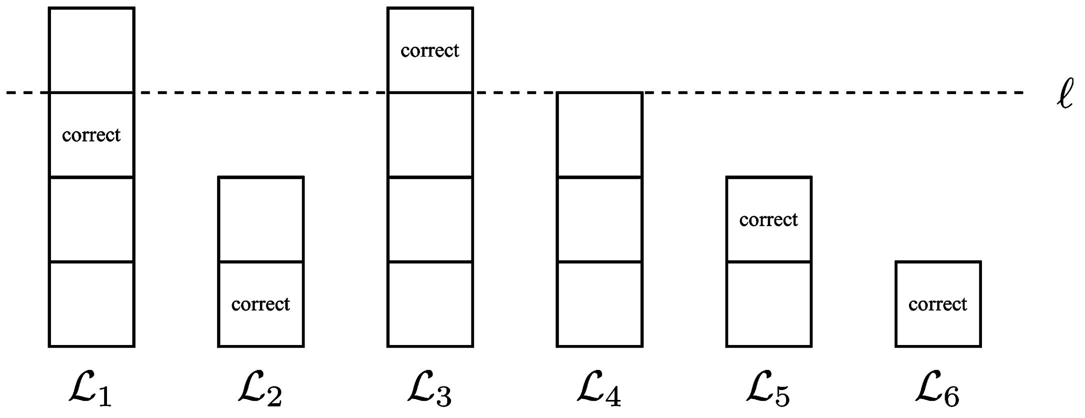

Fig. 1. Example of received packets.

received and authenticated are going to be useless anyway. On the other hand, if $\mathcal{O}$ injects a large number of forged packets then the main problem is making the communication channel more resistant against DoS attacks. Indeed, if $\mathcal{O}$ injects too many packets, then the receiver will have to buffer data much faster than he can authenticate it. This will rapidly result in a saturation of his storage capacity. In order to build our signature amortization scheme, we split the data stream into blocks of n packets: $P_1, \ldots, P_n$. Each of these blocks is located within the whole stream using a Block IDentifiers value BID.

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, when n augmented packets are sent into the network, at least $\lceil \alpha\, n \rceil$ of them are received and the total number of received packets does not exceed $\lfloor \beta\, n \rfloor$. This can be summarized as follows.

Definition IV.1: We say that a pair (α, β) of survival and flood rates is accurate to the network for a flow of n elements if:

1) Data are sent per block of n elements through the network.
2) For any block of n elements $\{E_1, \ldots, E_n\}$ emitted by the sender, the set of received packets $\{\widetilde{E}_1, \ldots, \widetilde{E}_\mu\}$ satisfies $\mu \leq \lfloor \beta\, n \rfloor$ and $|\{E_1, \ldots, E_n\} \cap \{\widetilde{E}_1, \ldots, \widetilde{E}_\mu\}| \geq \lceil \alpha\, n \rceil$. The second condition must be true for each receiver belonging to the communication group.

In this model, the parameter α represents the quality of the transmission medium. For instance, if fiber is used, then α will be extremely close to 1 whereas wireless networks will experience a smaller value for that coefficient due to the higher erasure rate for this technology. The parameter β represents the noxiousness of the adversary. A network where the flow of data can easily be influenced by an outsider will exhibit a large value for this parameter. Determining appropriate values (α, β) for a given network is not in the scope of this paper. Therefore, we assume that some preliminary study of the network topology has already determined the couple (α, β).

In the remaining of this paper, we assume that (α, β) is accurate to the network for a flow of n augmented packets.

B. Overview of the Construction

In this section, we give a general overview of our scheme. We need a collision resistant hash function h and a digital signature (KeyGen, $\mathrm{Sign}_{\mathrm{SK}}$, $\mathrm{Verify}_{\mathrm{PK}}$) which is secure against chosen message attacks.

1) Protocol at the Sender: From the n data packets $P_1, \ldots, P_n$, we want to construct n augmented packets $\mathrm{AP}_1, \ldots, \mathrm{AP}_n$ such that if at most $n - \lceil \alpha\ n \rceil$ of them are lost during transmission then the receiver can still recover all the P_i's. Thus, we need to encode these n packets using an $[n, \lceil \alpha\ n \rceil, n - \lceil \alpha\ n \rceil + 1]$ code. To perform this encoding, the size of elements forming the code's alphabet will be larger than the size of a data packet by a factor roughly equal to $\frac{1}{\alpha}$.

In order to provide nonrepudiation and to deal with bogus injections, we hash the codeword coordinates $C_1, \ldots, C_n$ (generated by the MDS code) and sign the concatenation $h(C_1)\| \cdots \|h(C_n)$ into σ. We encode $h(C_1)\| \cdots \|h(C_n)\|\sigma$ using the (λ, ℓ, L)-list recoverable code into $(\widetilde{C_1 \cdots C_n})$. We build the augmented packets as: $\forall i \in \{1, \ldots, n\}$ $\mathrm{AP}_i := \mathrm{BID}\|i\|C_i\|\widetilde{C}_i$.

2) Protocol at the Receiver: Upon reception of a list $\mathcal{R}$ of packets, the receiver stacks the candidate values for the $\widetilde{C}_i$'s as on Fig. 1.

Then, he deletes the columns where there are more than ℓ candidate values as depicted on Fig. 2.

He outputs the list of codewords which are consistent with the remaining stacks. He decodes these codewords which leads to the recovery of the block signature σ as well as the digests $h(C_1), \ldots, h(C_n)$. Thus, he can identify the correct C_i's as a part of packets in $\mathcal{R}$. According to the definition of α, there must be at least $\lceil \alpha\ n \rceil$ symbols from $C_1, \ldots, C_n$. Finally, he corrects the erasures using the MDS code and recovers the n data packets $P_1, \ldots, P_n$.

C. Formal Scheme

In this section, we describe a multicast authentication protocol using list recoverable code codes which is robust against packet loss and data injection. As in [14], [17]–[21], our technique allows any new user to join the communication group at any block boundary.

1) Parameters of the List Recoverable Code: Our construction requires a (λ, ℓ, L)-list recoverable code of dimension k and length n over the finite field with $2^{\tilde{q}}$ elements $\mathbb{F}_{2^{\tilde{q}}}$. Now, we discuss the relations those parameters have to satisfy.

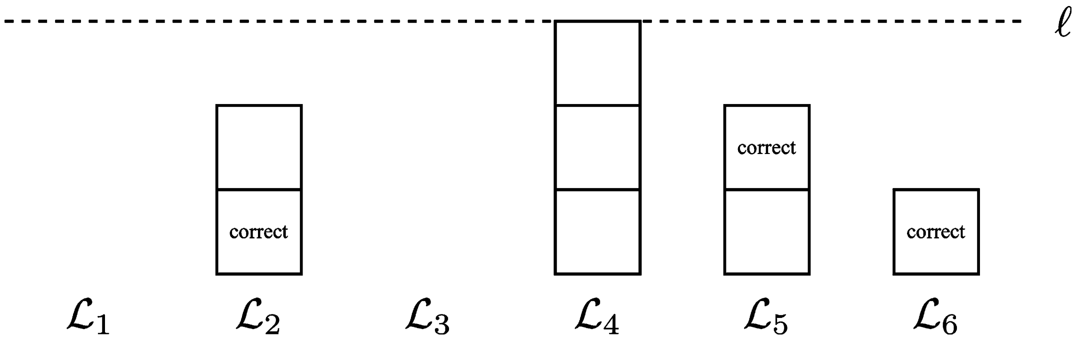

Fig. 2. Remaining candidates after column deletions.

The first point is the deletion process shown on Figs. 1 and 2. Since the pair of rates (α, β) is accurate, at least $\lceil \alpha\, n \rceil$ columns contain one correct element before deletions. At the same time, the opponent $\mathcal{O}$ injects at most $\lfloor \beta\, n \rfloor - \lceil \alpha\, n \rceil$ bogus packets. As any column can contain up to ℓ elements without being truncated, we deduce that the number of columns containing a correct element which can be deleted is at most

$$\left\lfloor \frac{\lfloor \beta\, n \rfloor - \lceil \alpha\, n \rceil}{\ell} \right\rfloor .$$

This bound is reached when the receiver collects elements for only $\lceil \alpha\, n \rceil$ columns while $\mathcal{O}$ exclusively pollutes those positions as well. The receiver gets no values for the other $n - \lceil \alpha\, n \rceil$ columns. Therefore, despite deletions of columns, each receiver is guaranteed that there will be at least

$$\lceil \alpha\, n \rceil - \left\lfloor \frac{\lfloor \beta\, n \rfloor - \lceil \alpha\, n \rceil}{\ell} \right\rfloor .$$

columns containing one correct element after deletions. In order to use the code, we must have

$$\lceil \alpha\, n \rceil - \left\lfloor \frac{\lfloor \beta\, n \rfloor - \lceil \alpha\, n \rceil}{\ell} \right\rfloor \geq \lambda\, n.$$

Secondly, the message to be encoded is $n\, \mathcal{H} + \mathcal{S}$ bits long before padding where $\mathcal{H}$ the digest bit length and $\mathcal{S}$ the bit length of a signature (see Step 3 of Algorithm 1). As the dimension of the code is k, we must have

$$\mathcal{H}\, n + \mathcal{S} \leq k\, \widetilde{q}.$$

Algorithm 1 Authenticator

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

/*Packet Encoding*/

1. Compute: $r = n\mathcal{P} \mod \lceil \alpha\, n \rceil$. Denote p as $p := (0 \text{ if } r = 0)$ or $(\lceil \alpha\, n \rceil - r$ otherwise). Write $P_1 || \cdots || P_n || 0^p$ as $M_1 || \cdots || M_{\lceil \alpha n \rceil}$ where each M_i is $q := \left\lceil \frac{n\mathcal{P}}{\lceil \alpha n \rceil} \right\rceil$ bits long. Encode the message $(M_1 \cdots M_{\lceil \alpha n \rceil})$ into the codeword $(C_1 \cdots C_n)$ using the MDS code over $\mathbb{F}_{2^q}$.

/*Signature Generation*/

2. Compute the coordinate digests $h(C_1), \dots, h(C_n)$ and construct the block signature σ as $\sigma = \mathrm{Sign}_{\mathrm{SK}}(h(\mathrm{BID} || \tau_{\mathrm{par}} || h(C_1) || \cdots || h(C_n)))$.

/*Construction of the Augmented Packets*/

3. Parse $h(C_1) || \cdots || h(C_n) || \sigma$ into $\widetilde{M_1} || \cdots || \widetilde{M_k}$ after padding. Encode the message $(\widetilde{M_1} \cdots \widetilde{M_k})$ into the codeword $(\widetilde{C_1} \cdots \widetilde{C_n})$ using the list recoverable code over $\mathbb{F}_{2^{\widetilde{q}}}$.

4. Build the augmented packets as

$$\forall i \in \{1, \dots, n\} \quad \mathrm{AP}_i := \mathrm{BID} || i || C_i || \widetilde{C_i}.$$

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

TABLE I
PUBLIC PARAMETERS FOR OUR SCHEME

n: Block length α: Survival rate	$\mathcal{P}$: Packet size (in bits) β: Flood rate
A list of irreducible polynomials over $\mathbb{F}_2$	
λ, ℓ, L: Parameters of the list recoverable code	
$k, \mathbb{F}_{2^{\widetilde{q}}}$: Dimension and field of the list recoverable code	

We deduce that a necessary and sufficient condition to use a (λ, ℓ, L)-list recoverable code of dimension k and length n over $\mathbb{F}_{2^{\widetilde{q}}}$ is

$$\begin{cases} \lceil \alpha\, n \rceil - \left\lfloor \frac{\lfloor \beta\, n \rfloor - \lceil \alpha\, n \rceil}{\ell} \right\rfloor \geq \lambda\, n \\ \mathcal{H}\, n + \mathcal{S} \leq k\, \widetilde{q} \end{cases} . \quad (1)$$

In the remaining of this paper, we assume that the list recoverable code satisfies (1).

2) Authentication Scheme: For our construction, we assume that α and β are rational numbers. Thus, we can represent them over a finite number of bits using their numerator and denominator. Table I summarizes the scheme parameters which are assumed to be publicly known.

The hash function h as well as Verify and PK are also assumed to be publicly known. We did not include them in Table I since they can be considered as general parameters. For instance, h can be SHA-256 while the digital signature can be a 1024-bit RSA signature [38]. Since h and the digital signature are publicly known, so are $\mathcal{H}$ and $\mathcal{S}$.

We denote by τ_{par} the tag representing the communication parameters, namely: $\tau_{\mathrm{par}} := n || \alpha || \beta || \mathcal{P}$. It is assumed that the list of polynomials contains a single polynomial $\mathcal{T}(X)$ per degree value $\deg(\mathcal{T}(X))$. Note that the elements of Table I uniquely determine τ_{par}. The sender processes the data stream as specified by Algorithm 1. The receivers process information using Algorithm 2.

Algorithm 2 Decoder

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

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

2. Rename the remaining elements as $\mathrm{AP}'_1,\ldots,\mathrm{AP}'_N$ and write each element as: $\mathrm{AP}'_i = \mathrm{BID} \| j_i \| C'_{j_i} \| \widetilde{C}'_{j_i}$ where $j_i \in \{1,\ldots,n\}$.

/*Signature Verification*/

3. Initialize $\widetilde{L}_i = \emptyset$ for $i \in \{1,\ldots,n\}$. For $i \in \{1,\ldots,N\}$, include $\widetilde{C}'_{j_i}$ into $\widetilde{L}_{j_i}$. For $i \in \{1,\ldots,N\}$, if $|\widetilde{L}_{j_i}| > \ell$ then re-set $\widetilde{L}_{j_i} = \emptyset$.

4. Recover the list of codewords $\widetilde{\mathcal{C}}_1,\ldots,\widetilde{\mathcal{C}}_\mu$ agreeing with the lists $\widetilde{L}_1,\ldots,\widetilde{L}_n$.

5. Initialize $h_i = \emptyset$ for $i \in \{1,\ldots,n\}$. Set $j = 0$. While (the list has not been exhausted) and (the signature has not been verified) do:

5.1 Decode $\widetilde{\mathcal{C}}_j$ into $\widetilde{\mathcal{M}}_j$. Parse the latter as: $\widetilde{C}_{j,1} \| \cdots \| \widetilde{C}_{j,n} \| \widetilde{\sigma}$ after removing the pad where each $\widetilde{C}_{j,i}$ is $\mathcal{H}$ bits long.

5.2 If $\mathrm{Verify}_{\mathrm{PK}}(h(\mathrm{BID} \| \tau_{\mathrm{par}} \| \widetilde{C}_{j,1} \| \cdots \| \widetilde{C}_{j,n}), \widetilde{\sigma}) = \mathrm{TRUE}$ then set $h_i = \widetilde{C}_{j,i}$ for $i \in \{1,\ldots,n\}$ and break the loop. Otherwise, increase $j$ by 1.

6. If the signature has not been verified then the algorithm stops.

/*Packet Recovery*/

7. Set $\mathcal{C}'_i = \emptyset$ for $i \in \{1,\ldots,n\}$. For $i \in \{1,\ldots,N\}$, if $h(C'_{j_i}) = h_\xi$ then set $\mathcal{C}'_\xi = C'_{j_i}$.

8. If $(\mathcal{C}'_1 \cdots \mathcal{C}'_n)$ has less than $\lceil \alpha n \rceil$ nonerased coordinates then the algorithm stops. Otherwise, compute $q$ as in Step 1 of Authenticator and correct the erasures over $\mathbb{F}_{2^q}$ of the MDS codeword $(\mathcal{C}'_1 \cdots \mathcal{C}'_n)$.

9. Denote $(M'_1 \cdots M'_n)$ the corresponding message. Parse it as $P'_1 \| \cdots \| P'_n$ where each $P'_i$ is $\mathcal{P}$ bits long after removing the pad.

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

V. Analysis of the Protocol

A. *Security of the Scheme*

Similar to [14], [18], [20], [21], we give the following definition:

Definition V.1: A collection of algorithms (KeyGen, Authenticate, Decode) constitutes a secure and (α,β)-correct probabilistic multicast authentication scheme if no PPT opponent $\mathcal{O}$ can win with a non-negligible probability the following game:

i) A key pair (SK,PK) is generated by KeyGen.
ii) $\mathcal{O}$ is given: (a) the public key PK and (b) oracle access to Authenticate (but $\mathcal{O}$ can only issue at most one query with the same block identification tag BID).
iii) $\mathcal{O}$ outputs $(\mathrm{BID}, n, \alpha, \beta, \rho, \mathcal{P}, \mathrm{RP})$.

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

a) (violation of the correctness property) $\mathcal{O}$ succeeds to construct RP such that even if it contains $\lceil \alpha\, n \rceil$ packets (amongst a total not exceeding $\lfloor \beta\, n \rfloor$ elements) of some authenticated packet set AP for block identification tag BID and parameters n, α, β, ρ, $\mathcal{P}$, the decoder fails to authenticate all the correct packets.
b) (violation of the security property) $\mathcal{O}$ succeeds to construct RP such that the decoder outputs $\{P'_1,\ldots,P'_n\}$ that was never authenticated by Authenticate for the value BID and parameters n, α, β, ρ, $\mathcal{P}$.

We have the following security result for our authentication protocol. Its demonstration can be found in Appendix I.

Theorem V.2: The authentication scheme (KeyGen, Authenticator, Decoder) is secure and (α,β)-correct.

B. *Efficiency of the Scheme*

1) Data Recovery: We will now demonstrate that our scheme enables any receiver to recover the n data packets (as in [18], [20], [21]). The following theorem is another way of expressing Theorem V.2.

Theorem V.3: Given the scheme (KeyGen, Authenticator, Decoder), for any BID, each receiver recovers the n original data packets $P_1,\ldots,P_n$.

Proof: Let BID be the current block value. Because the pair of rates (α,β) is accurate, at least $\lceil \alpha\, n \rceil$ of the original elements $\mathrm{AP}_1,\ldots,\mathrm{AP}_n$ are received by the receiver amongst a total of no-more than $\lfloor \beta\, n \rfloor$ elements. Thus, the demonstration of Theorem V.2 shows that Decoder returns $P_1,\ldots,P_n$ since the digital signature is secure against chosen message attack and the hash function is collision resistant. ∎

2) Signature Verification Cost: In this section, we derive an upper bound on the number of signature verification queries performed by the receiver for each block of packets.

Theorem V.4: Given the scheme (KeyGen, Authenticator, Decoder), for any BID, the number of signature verifications to be performed by each receiver is upper bounded by L.

Proof: Let BID be the current block value. Because the pair of rates (α,β) is accurate, at least $\lceil \alpha\, n \rceil$ of the original elements $\mathrm{AP}_1,\ldots,\mathrm{AP}_n$ are received by the receiver amongst a total of up to $\lfloor \beta\, n \rfloor$ elements. Thus, the proof of Theorem V.2 shows that one of the elements from the list $\{\widetilde{\mathcal{C}}_1,\ldots,\widetilde{\mathcal{C}}_\mu\}$ constructed at Step 4 of Algorithm 2 is equal to the original string $\widetilde{M}_1 \| \cdots \| \widetilde{M}_k$. Thus, there are at most μ signature verification queries. Due to the design of the list recoverable code, we have: $\mu \leq L$. ∎

3) Packet Overhead: The augmented packets are written as

$$\mathrm{BID}||i||C_i||\tilde{C}_i.$$

The bit size of the element $\tilde{C}_i$ is $\tilde{q}$ and C_i is slightly larger than P_i since it is $\left\lceil \frac{\mathcal{P}\, n}{\lceil \alpha n \rceil} \right\rceil$ bits long. So, our packet overhead is

$$\left\lceil \frac{\mathcal{P}\, n}{\lceil \alpha n \rceil} \right\rceil - \mathcal{P} + \tilde{q}\ \text{bits}.$$

Notice that the difference $\left\lceil \frac{\mathcal{P}\, n}{\lceil \alpha n \rceil} \right\rceil - \mathcal{P}$ comes from the fact that we use a code correcting up to a fraction $1-\alpha$ of erasures.

4) Codes: Our authentication protocol uses coding techniques for two different purposes.

- **Data Recovery.** This occurs at Step 8 of Decoder where the goal is to reconstruct the n symbols $C_1, \ldots, C_n$ generated at Step 1 of Authenticator. This code is only utilized to correct erasures. This part of our authentication protocol is similar to dealing with an erasure channel having reliable symbol transmission rate α for an alphabet of size 2^q where q is roughly $\frac{\mathcal{P}}{\alpha}$. Implementing this section of the protocol simply requires to use a code which can correct up to a $(1-\alpha)$-fraction of erasures.
- **Data Authentication.** This occurs at Step 4 and Step 5 of Decoder where the goal is to reconstruct the signature generated at Step 2 of Authenticator. We use the list decoding property of the code for an alphabet of size $2^{\tilde{q}}$. The value of $\tilde{q}$ depends on the pad needed to encode $h(C_1)||\cdots||h(C_n)||\sigma$ (see Section VI for an illustration using RS codes). It should be noticed that, if the pad is not taken into account, the length of the string to be encoded is $n\,\mathcal{H} + \mathcal{S}$ bits which is independent of the packet bit length $\mathcal{P}$.

VI. Application of our Construction Using Reed–Solomon Codes

In Section IV, we presented a theoretical approach based on list recoverable codes to ensure stream authentication. In this section, we illustrate that construction using RS codes.

A. Performing List Recovering of Reed–Solomon Codes

Let $\mathbb{F}_r$ be the finite field with r elements and denote by $\mathbb{F}_r[X]$ the ring of univariate polynomials over $\mathbb{F}_r$. Consider $x_1, \ldots, x_n$ as n distinct elements of $\mathbb{F}_r$. We define the following mapping:

$$\begin{aligned} \mathrm{Eval}:\quad & \mathbb{F}_r[X] \to \mathbb{F}_r^n \\ & P(X) \mapsto (P(x_1), \ldots, P(x_n)). \end{aligned}$$

Definition VI.1 ([49]): The Reed–Solomon (RS) code of dimension k and length n over $\mathbb{F}_r$ is the following set of n-tuples (codewords):

$$\{\mathrm{Eval}(P(X)) : P(X) \in \mathbb{F}_r[X] \text{ and } \deg(P(X)) < k\}.$$

Remark VI.2: It is clear that recovering $P(X)$ corresponds to solving an instance of the noisy PRP. An important fact is that Poly-Reconstruct directly outputs a list of candidates for the original message $P(X)$. As a consequence, Poly-Reconstruct enables to merge Step 5.1 and Step 5.2 of Algorithm 2.

Similar to [14], [18], [21], we use an RS code of length n and dimension $\rho\, n + 1$. Now, we need to determine the parameter ρ so that Poly-Reconstruct can be used after deletion of columns.

Let $\mathcal{D}_\ell$ be the maximum number of columns which can be deleted. Since the pair of rates (α, β) is accurate, at most $\lfloor \beta\, n \rfloor$ elements are collected by each receiver. Since a column is not deleted if it contains up to ℓ candidate values, we have

$$\mathcal{D}_\ell = \left\lfloor \frac{\lfloor \beta\, n \rfloor}{\ell + 1} \right\rfloor.$$

We need to have at least one correct element after deletions. In the worst case, there are only $\lceil \alpha\, n \rceil$ original elements reaching the receiver. Thus, the number of these elements remaining after deletion is at least $\max(\lceil \alpha\, n \rceil - \mathcal{D}_\ell, 0)$. We need to have

$$\lceil \alpha\, n \rceil - \mathcal{D}_\ell \geq 1. \tag{2}$$

We assume that (2) holds. The degree of the polynomial used for the RS code is $\rho\, n$. If $d (\in \{0, \ldots, \mathcal{D}_\ell\})$ columns are deleted, then we have at least $\lceil \alpha\, n \rceil - d$ correct elements amongst at most $\lfloor \beta\, n \rfloor - d\,(\ell+1)$ points. Remark that $\lfloor \beta\, n \rfloor - d\,(\ell+1) > 0$ since (2) holds and $d \leq \mathcal{D}_\ell$. As a consequence, we require

$$\forall d \in \{0, \ldots, \mathcal{D}_\ell\} \quad (\lceil \alpha\, n \rceil - d)^2 > \rho\, n\,(\lfloor \beta\, n \rfloor - d\,(\ell+1)).$$

which can also be written as

$$\rho < \min_{d \in \{0, \ldots, \mathcal{D}_\ell\}} \left(\frac{(\lceil \alpha\, n \rceil - d)^2}{n\,(\lfloor \beta\, n \rfloor - d\,(\ell+1))} \right).$$

In order to study this minimum, we consider the following real mapping

$$\begin{aligned} f : [0, \mathcal{D}_\ell] & \to \mathbb{R}^+ \\ d & \mapsto \frac{(\lceil \alpha\, n \rceil - d)^2}{n\,(\lfloor \beta\, n \rfloor - d\,(\ell+1))}. \end{aligned}$$

In particular, we must have $\rho < f(0)$. It should be noticed that the value $\tilde{\rho}$ used in [14], [18], [21] is at most $f(0)$. In order to obtain the same bound, we must have $f(0) = \min_{d \in [0, \mathcal{D}_\ell]} f(d)$. The mapping f is differentiable over its domain and we get the following result:

$$f'(d) > 0 \Longleftrightarrow d > 2\,\frac{\lfloor \beta\, n \rfloor}{\ell + 1} - \lceil \alpha\, n \rceil.$$

In order to have $f(0) = \min_{d \in [0, \mathcal{D}_\ell]} f(d)$, we must have

$$0 \geq 2\,\frac{\lfloor \beta\, n \rfloor}{\ell + 1} - \lceil \alpha\, n \rceil.$$

which is equivalent to: $\ell \geq \left\lceil \frac{2\,\lfloor \beta\, n \rfloor}{\lceil \alpha\, n \rceil} \right\rceil - 1$ as ℓ is an integer. This lower bound is denoted $\ell_{\min}$.

We showed that if (2) held then choosing ℓ no smaller than $\ell_{\min}$ guaranteed to run Poly-Reconstruct while $f(0) = \min_{d \in [0, \mathcal{D}_\ell]} f(d)$. Now, we would like to see whether any such a ℓ verifies (2). It is easy to see that we have

$$\ell \geq \ell_{\min} \Longrightarrow \lceil \alpha\, n \rceil - \frac{\lfloor \beta\, n \rfloor}{\ell + 1} \geq \frac{\lceil \alpha\, n \rceil}{2}.$$

The parameter α is the survival rate of the network for a flow n. Thus, without loss of generality, one can assume that $\lceil \alpha\, n \rceil \geq 2$. Otherwise, it would mean that some receivers may get no more than one original packet from the sender which is not a realistic network assumption. As a consequence, we get

$$\ell \geq \ell_{\min} \Longrightarrow \lceil \alpha\, n \rceil - \mathcal{D}_\ell \geq 1.$$

Thus, any $\ell \geq \ell_{\min}$ ensures that at least 1 original element is not deleted when columns are removed. More precisely, we showed that at least $\frac{\lceil \alpha\, n \rceil}{2}$ original elements remain after those column deletions.

Theorem VI.3: Let ℓ be any integer no smaller than $\ell_{\min}$. The previous RS code is $\left(\frac{\lceil \alpha\, n \rceil - d}{n}, \ell, U(n,\ell,d)\right)$-list recoverable from any set of at most $\lfloor \beta\, n \rfloor - d\,(\ell+1)$ points where d is any integer in $\{0,\dots,\mathcal{D}_\ell\}$ with: $U(n,\ell,d) := \min(\lfloor U_1(n,\ell,d) \rfloor, \lfloor U_2(n,\ell,d) \rfloor)$ as defined (see the equation at the bottom of the page).

The proof of the previous theorem can be found in Appendix II.

Remark VI.4: The reader may notice that the properties of the RS code described above do not verify (1) as we have

$$\lceil \alpha\, n \rceil - \left\lfloor \frac{\lfloor \beta\, n \rfloor - \lceil \alpha\, n \rceil}{\ell} \right\rfloor < \frac{\lceil \alpha\, n \rceil - d}{n}\, n.$$

for some $d \in \{0,\dots,\mathcal{D}_\ell\}$. Thus, this RS code does not seem to be usable for our construction. Hopefully, this is not the case for the following reason. In Section IV, the protocol requires to use a (λ,ℓ,L)-list recoverable code whose parameters verify (1). Theorem VI.3 does not claim that the RS code is $\left(\frac{\lceil \alpha\, n \rceil - d}{n}, \ell, U(n,\ell,d)\right)$-list recoverable in the sense of Definition III.6. Indeed, in that definition, it is allowed to have up to ℓ candidate values for each of the n positions while the recovery of the RS code is guaranteed provided that there are no more than $\lfloor \beta\, n \rfloor - d\,(\ell+1)$ values in total. The latter condition is due to the necessity of running Poly-Reconstruct. However, this restriction does not weaken the security of the scheme as the list output by Poly-Reconstruct will still contain $h(P_1)||\cdots||h(P_n)||\sigma$ due to its consistency as an error correcting decoder.

TABLE II
PACKET OVERHEAD WHEN $n = 1000$ FOR $\mathcal{P} = 512$ (LEFT) $\mathcal{P} = 4096$ (RIGHT)

	(α,β)			
	$(0.5, 1.1)$	$(0.5, 1.25)$	$(0.5, 1.5)$	$(0.5, 2)$
10%	11345|14929	12752|16336	15061|18645	19551|23135
30%	4228|7812	4726|8310	5552|9136	7188|10772
50%	2755|6339	3057|6641	3560|7144	4560|8144
70%	2118|5702	2335|5919	2697|6281	3417|7001
90%	1763|5347	1933|5517	2215|5799	2777|6361
	(α,β)			
	$(0.75, 1.1)$	$(0.75, 1.25)$	$(0.75, 1.5)$	$(0.75, 2)$
10%	5101|6296	5759|6954	6847|8042	8996|10191
30%	1836|3031	2061|3256	2436|3631	3182|4377
50%	1173|2368	1309|2504	1535|2730	1986|3181
70%	888|2083	985|2180	1147|2342	1470|2665
90%	729|1924	805|2000	931|2126	1183|2378
	(α,β)			
	$(0.8, 1.1)$	$(0.8, 1.25)$	$(0.8, 1.5)$	$(0.8, 2)$
10%	4471|5367	5052|5948	6015|6911	7917|8813
30%	1593|2489	1791|2687	2121|3017	2778|3674
50%	1009|1905	1129|2025	1328|2224	1725|2621
70%	758|1654	844|1740	986|1882	1271|2167
90%	618|1514	685|1581	796|1692	1018|1914
	(α,β)			
	$(0.9, 1.1)$	$(0.9, 1.25)$	$(0.9, 1.5)$	$(0.9, 2)$
10%	3501|3900	3964|4363	4731|5130	6251|6650
30%	1216|1615	1373|1772	1634|2033	2156|2555
50%	754|1153	848|1247	1006|1405	1321|1720
70%	555|954	623|1022	736|1135	961|1360
90%	445|844	497|896	585|984	761|1160

B. Comparison to PRP-Based Broadcast Authentication Schemes

As discussed in Section VI-A, using the minimal value $\ell_{\min}$ involves that our construction has the same packet overhead as [14], [18], [21]. More precisely, our packet overhead is

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

where ξ is the smallest element of $\mathbb{N}$ such that

$$\left\lceil \frac{\mathcal{H}\, n + \mathcal{S} + \xi}{\rho\, n + 1} \right\rceil \geq \lceil \log_2 n \rceil . \tag{3}$$

Table II depicts the packet overhead for this construction for $n = 1000$ and for the values of (α,β) as used in [18], [20], [21]. The parameter ρ has been chosen so that $\rho\, n$ represents 10%, 30%, 50%, 70% and 90% of the threshold value $\frac{\alpha^2}{\beta}$ as in [18], [21]. As in [11], we chose two different values for the bit size $\mathcal{P}$ of data packets, namely: 512 bits and 4096 bits. These packet size values were used by Pannetrat and Molva to illustrate the data flow when (1) collecting traffic information from sensors

$$U_1(n,\ell,d) = \frac{1 - \frac{d}{n}}{\rho} \left(1 + \frac{\rho\, \left(\beta - \frac{\ell+1}{n}\, d\right)}{\left(\alpha - \frac{d}{n}\right)^2 - \rho\, \left(\beta - \frac{\ell+1}{n}\, d\right)} + \frac{1}{n\, \sqrt{\left(\alpha - \frac{d}{n}\right)^2 - \rho\, \left(\beta - \frac{\ell+1}{n}\, d\right)}} \right) - \frac{1}{\rho\, n}$$

$$U_2(n,\ell,d) = \left(1 - \frac{d}{n}\right) \left(\frac{1}{\rho} + \frac{\left(\beta - \frac{\ell+1}{n}\, d\right) + \sqrt{\left(\beta - \frac{\ell+1}{n}\, d\right)^2 + \frac{4}{\rho^2\, n^2}\, \left(\left(1 - \frac{d}{n}\right) - \rho \left(\alpha - \frac{d}{n}\right)\right)}}{2\, \left[\left(\alpha - \frac{d}{n}\right)^2 - \rho\, \left(\beta - \frac{\ell+1}{n}\, d\right)\right]} \right) - \frac{1}{\rho n}$$

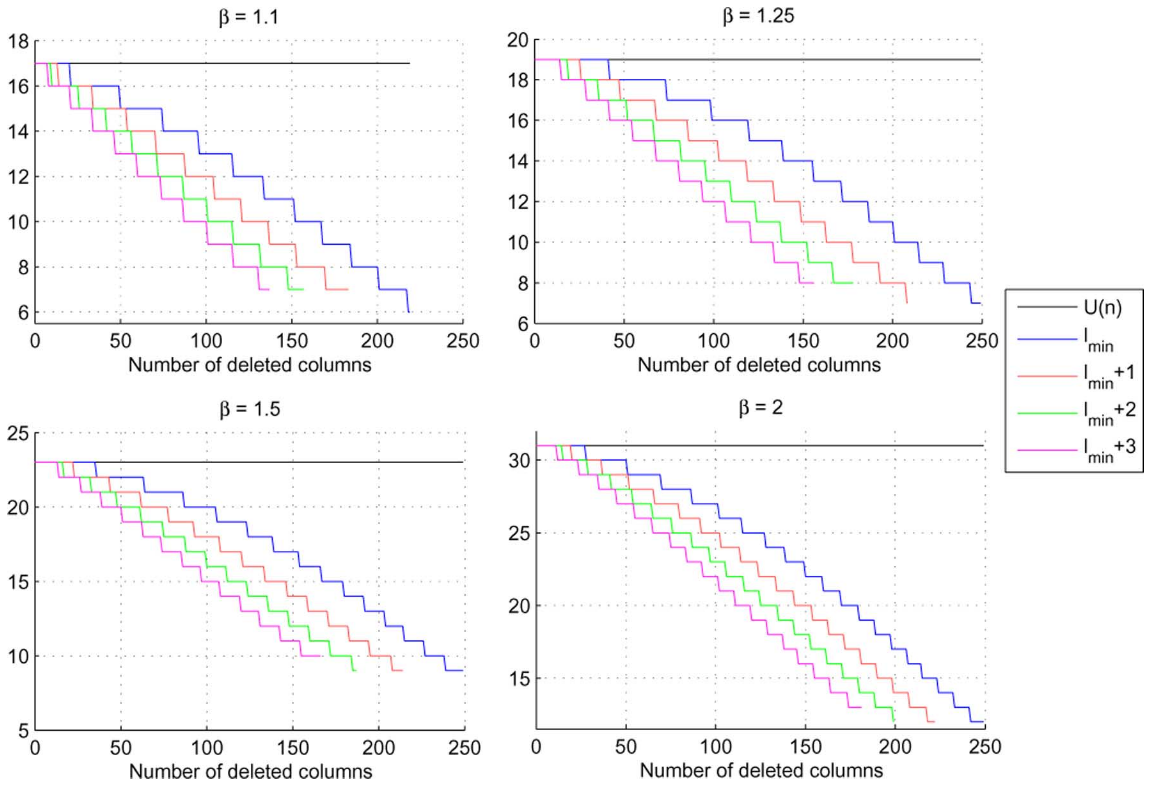

Fig. 3. Comparison of Bounds when $\alpha = 0.5$.

distributed over the street of a city (512 bits) and (2) performing real-time video broadcasting over the Internet (4096 bits). We used SHA-256 as a hash function and a 1024-bit RSA signature scheme (i.e., $\mathcal{H} = 246$ and $\mathcal{S} = 1024$).

We will now demonstrate that our scheme exhibits a smaller upper bound on the number of signature verification queries to be performed by the receiver. Note that [17], [20] are generalizations of [18] to multiple blocks. Thus, our approach will trivially lead to similar benefits when used over several blocks of data as well.

It has been demonstrated in [18] that an upper bound for the number of signature verification queries per block for [14], [18], [21] was $U(n) := \min(\lfloor U_1(n) \rfloor, \lfloor U_2(n) \rfloor)$ where

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

The demonstration of the following theorem can be found in Appendix III.

Theorem VI.5: Let ℓ be any integer no smaller than $\ell_{\min}$. We have

$$\forall d \in \{0, \ldots, \mathcal{D}_\ell\} \quad U(n, \ell, d) \le U(n).$$

The previous theorem shows that, whatever the number of deleted columns is, our list recoverable code approach leads to a better upper bound on the number of signature verification queries for *each* receiver. It should be noticed that the complexity cost of running Poly-Reconstruct on the set of nondeleted points is the same as using that algorithm on the whole set (having at most $\lfloor \beta\, n \rfloor$ elements) as in [14], [18], [21]. Thus, our deletion process does not increase the computational complexity at the receiver.

We illustrate the theoretical bounds comparison from Theorem VI.5 with the same values for n and the couple (α, β) as in Table II. The results are depicted as Figs. 3–6.

In all graphs, we can see that the difference between $U(n)$ and our bound $U(n, \ell, d)$ increases as a function of the number of truncated columns. This tends to imply that the column removal process has an impact on the list size and, by repercussion, on the number of signature queries performed by the receiver which was our primary purpose in introducing this technique. Also, we notice that the difference between the two upper bounds increases with ℓ when the number of truncated columns is fixed. This fact must not be misinterpreted. Indeed, it may be appealing to deduce that choosing a large value for ℓ is a good approach. However, increasing ℓ means that the truncation condition is relaxed. This can lead to reducing the number of columns to be discarded which, in turn, may worsen the bound. A trivial illustration of this fact is to consider $\ell = \lfloor \beta\, n \rfloor$ since no truncations are ever to occur in this situation.

C. Analysis of Polyreconstruct in the Context of Broadcast Authentication

1) Interpolation Multiplicity: One of the drawbacks of the previous works based on PolyReconstruct [14], [17]–[21], [29]

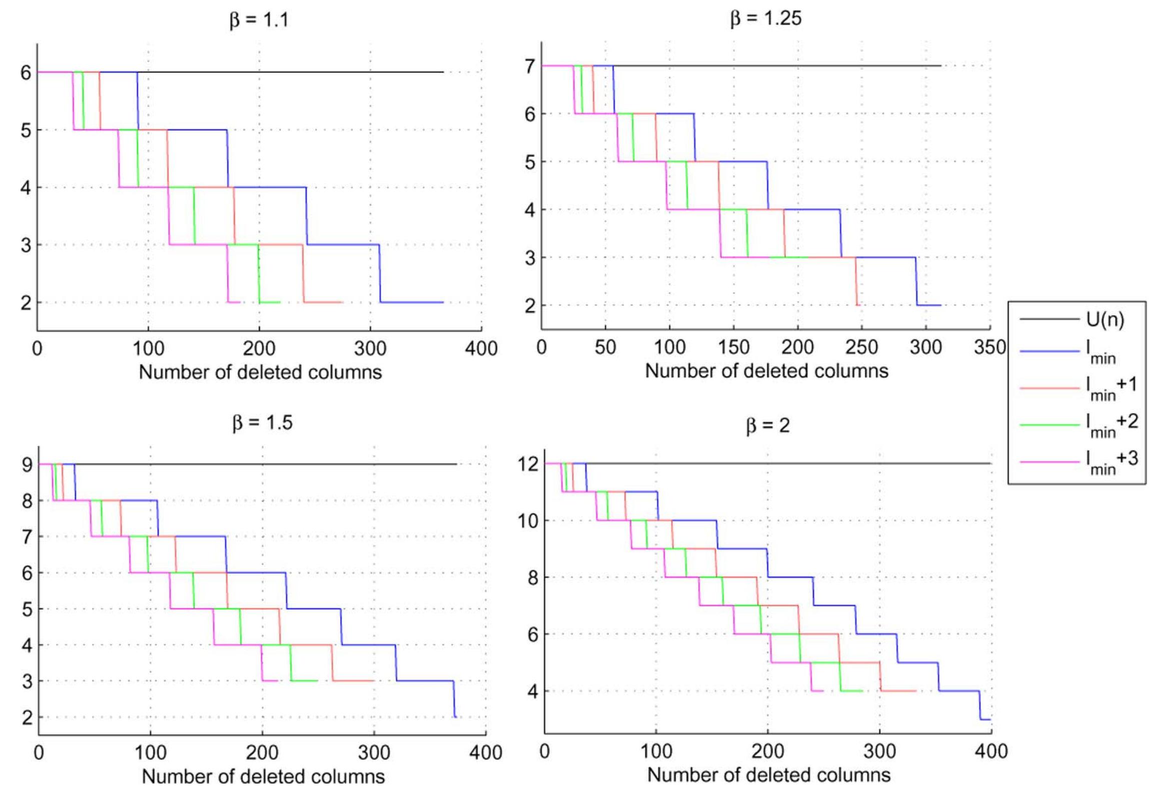

Fig. 4. Comparison of Bounds when $\alpha = 0.75$.

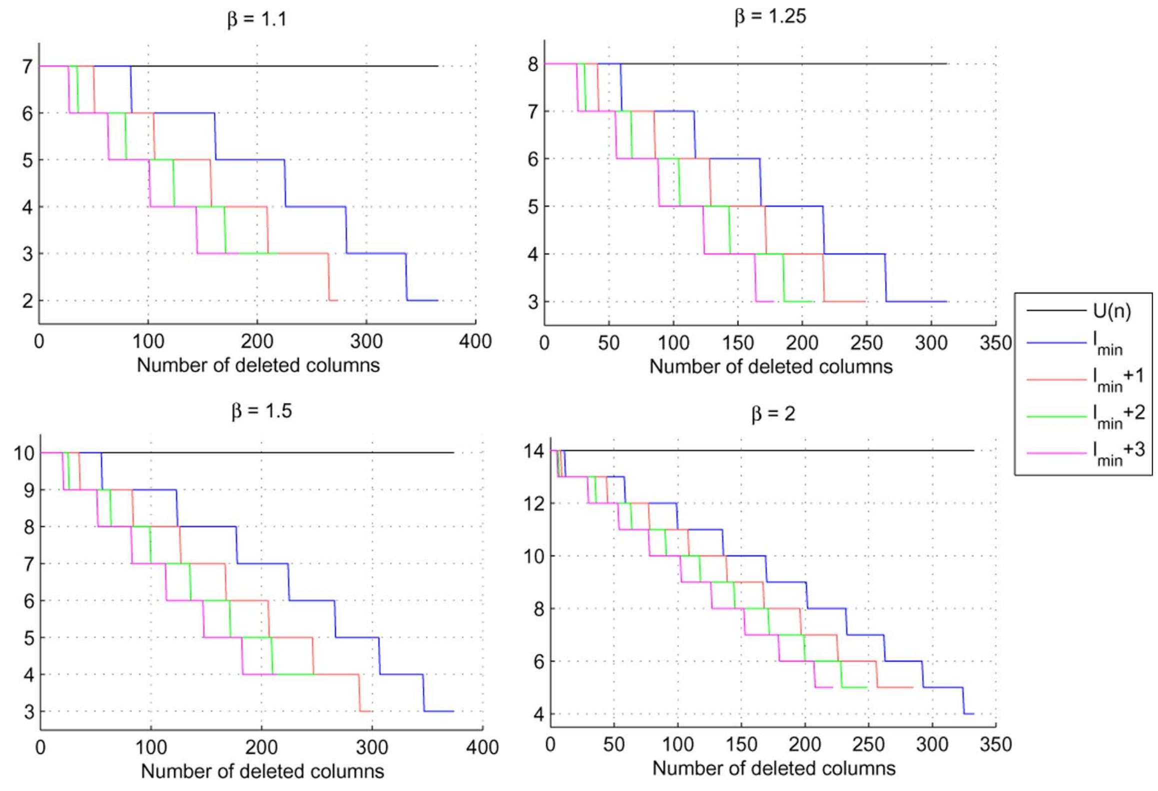

Fig. 5. Comparison of Bounds when $\alpha = 0.8$.

is that no information is given concerning the choice of the interpolation multiplicity m which is an essential parameter of the reconstruction algorithm. A large value will ensure the proper execution of the algorithm but, since its running time is $O(N^2m^4)$, one needs to choose a value as small as possible to preserve the practical efficiency of the whole authentication algorithm. Based on the work done by McEliece [34], we provide explicit choices of m to be used in our multicast setting. A more de-

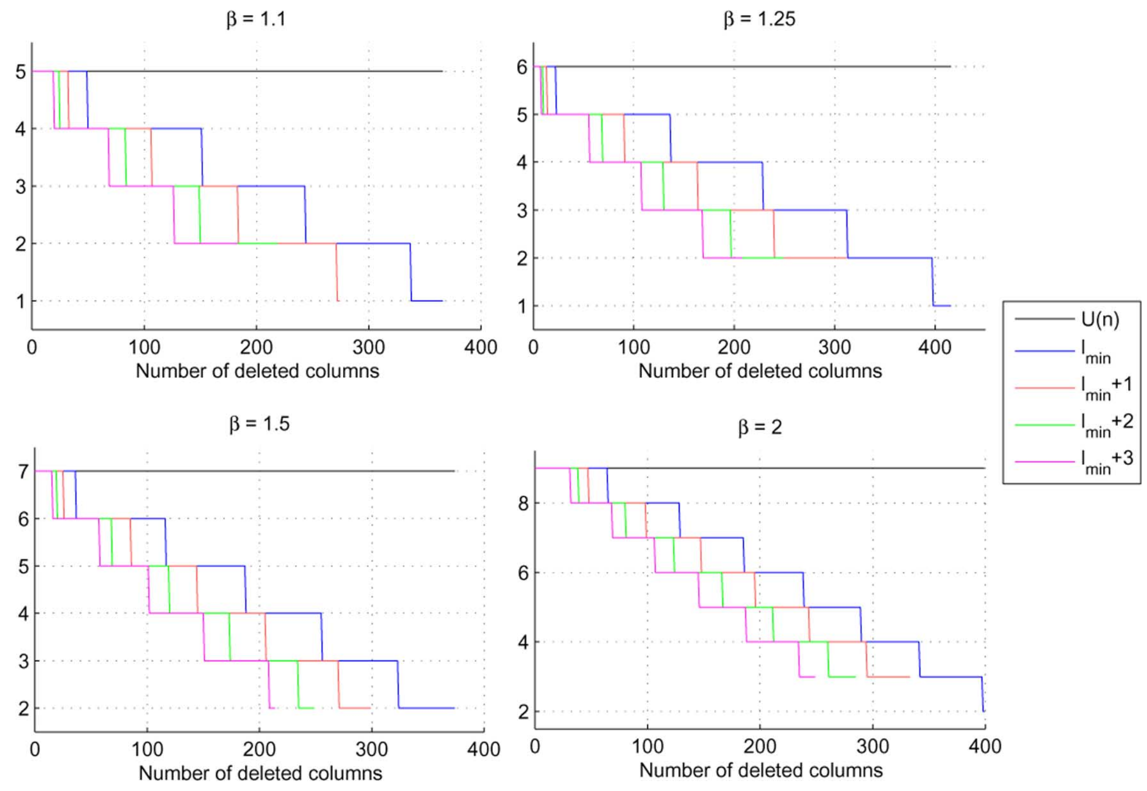

Fig. 6. Comparison of Bounds when $\alpha = 0.9$.

tailed analysis on the general behavior of PolyReconstruct can be found in [34]. We first recall the following two results from [34] using the notations of Section III-C.

Theorem VI.6 ([34]): The list output by PolyReconstruct contains every polynomial of degree at most D such that $T \geq K_m$. Furthermore, the number of polynomials in this list is at most L_m where

$$K_m = \min\left\{K : \mathrm{Ind}(X^{m\,K}) > n\binom{m+1}{2}\right\}$$
$$L_m = \left\lfloor \sqrt{\frac{N}{D}\,m\,(m+1) + \left(\frac{D+2}{2\,D}\right)^2} - \frac{D+2}{2\,D} \right\rfloor.$$

with $\mathrm{Ind}(X^{m\,K})$ being the index of the monomial $X^{m\,K}$ in the $(1, D)$-revlex monomial order.

Remark VI.7: Apart from reducing the running time of PolyReconstruct, choosing a small m also reduces the size of the list of polynomials since $m \mapsto L_m$ is an increasing mapping when N and D are constant. This will be particularly valuable in our multicast context as, in the worst case, the receiver needs to perform one signature verification query per element of the list (see Step 5.2 of Algorithm 2).

Unfortunately, computing the exact value of K_m is not simple. However, we have the following property.

Proposition VI.8 ([34]): We have

$$\left\lfloor \sqrt{D\,N\,\frac{m+1}{m}} - \frac{D}{2\,m} \right\rfloor + 1 \leq K_m \leq \left\lfloor \sqrt{D\,N\,\frac{m+1}{m}} \right\rfloor.$$

Given our network model, each participant collects at least $\lceil \alpha\, n \rceil$ genuine data packets. Therefore, we need to determine the smallest interpolation multiplicity $m_{\min}$ such that: $K_{\min} \geq \lceil \alpha\, n \rceil$ when $D = \rho\, n$ $\left(\text{for } \rho \in \left(0, \frac{\alpha^2}{\beta}\right)\right)$.

Let m be a positive integer. Using Proposition VI.8, ($K_m \leq \lceil \alpha\, n \rceil$) holds provided that

$$\left\lfloor \sqrt{\rho\, n\, \lfloor \beta\, n \rfloor\, \frac{m+1}{m}} \right\rfloor \leq \lceil \alpha\, n \rceil. \tag{4}$$

The previous inequality is verified as soon as we have

$$\sqrt{\rho\, n\, \beta\, n\, \frac{m+1}{m}} \leq \alpha\, n.$$
$$(4) \Longleftrightarrow \rho\,\beta\, n^2 \left(1 + \frac{1}{m}\right) \leq (\alpha\, n)^2 \Longleftrightarrow \frac{1}{m} \leq \frac{\alpha^2}{\rho\,\beta} - 1.$$

In our multicast settings, we have $\rho \in \left(0, \frac{\alpha^2}{\beta}\right)$. Thus, $\frac{\alpha^2}{\rho\,\beta} - 1$ is positive and we get:

$$(4) \Longleftrightarrow m \geq \frac{1}{\frac{\alpha^2}{\rho\,\beta} - 1}.$$

We denote $\widetilde{m}$ the following integer:

$$\widetilde{m} := \left\lceil \frac{1}{\frac{\alpha^2}{\rho\,\beta} - 1} \right\rceil.$$

By construction: $\widetilde{m} \geq m_{\min}$. From the previous analysis, we deduce the following theorem.

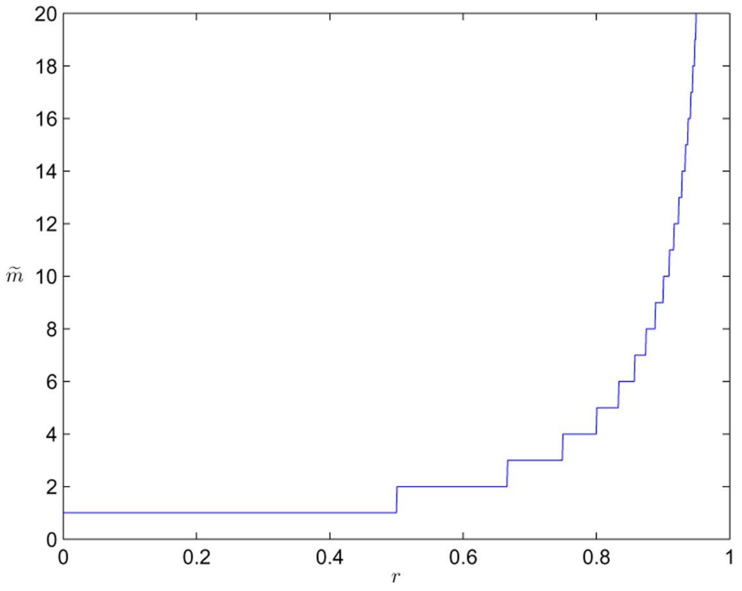

Fig. 7. Variations of the upper bound $\widetilde{m}$ on the interpolation multiplicity.

Theorem VI.9: Given an accurate pair of rates (α, β) for a flow of n elements, any receiver can use PolyReconstruct with interpolation multiplicity $\widetilde{m}$.

The important point to be noted is that $\widetilde{m}$ is valid for *any* receiver. A priori, this is not the case for $m_{\min}$ as $m_{\min}$ depends on the number of genuine packets the participant running PolyReconstruct has collected. The parameter $\widetilde{m}$ only depends on the general settings of the scheme: α, β, ρ.

Remark VI.10: The mapping $\rho \mapsto \widetilde{m}$ is increasing. Thus, choosing ρ "close" to 0 will reduce the interpolation multiplicity needed for PolyReconstruct. At the same time, the bound on the size of the list output by this algorithm will also be smaller since L_m is increasing as a function of m (and thus as a function of ρ). The drawback with this approach is related to the packet overhead of the authentication scheme. The "closer" to 0 the parameter ρ gets, the larger the packet overhead becomes since: $\left\lceil \frac{\mathcal{H}\, n+\mathcal{S}+\xi}{\rho\, n+1} \right\rceil \geq \frac{\mathcal{H}\, n+\mathcal{S}}{\rho\, n+1}$.

We now study the value $\widetilde{m}$. Denote r the real number such that: $\rho = r\, \frac{\alpha^2}{\beta}$. By construction: $r \in (0,1)$. We can rewrite $\widetilde{m}$ as

$$\widetilde{m} = \left\lceil \frac{1}{\frac{1}{r} - 1} \right\rceil .$$

Fig. 7 represents the graph of $r \mapsto \widetilde{m}$. As $\lim_{r \to 1} \widetilde{m} = +\infty$, we only drew the graph for r up to 0.95 to have a representative figure. An important fact is that $\widetilde{m}$ only depends on the value of r. Thus, the graph is valid for *any* pair of survival and flood rates (α, β).

Remark VI.11: For any $r \in (0, \frac{1}{2}]$, we have $\widetilde{m} = 1$. In such cases, PolyReconstruct is the original Sudan algorithm [50].

2) New Bound on the List Size: Based on the settings of Theorem VI.6, we rewrite L_m as $L(N, D, m)$ to emphasize the fact that the bound depends and N, D, m. In our multicast context, we have $D = \rho\, n$. The value $L(N, \rho\, n, m_{\min})$ is the bound on the list size for a participant receiving a total of N packets (including both genuine and forged ones). Like $m_{\min}$, the value $L(N, \rho\, n, m_{\min})$ depends on each participant since any two receivers $\mathcal{R}$ and $\mathcal{R}'$ will a priori receive a different amount of packets N and N'. However, to evaluate the global efficiency of the construction, we need to construct a bound valid for all receivers. In the worst case, a participant can obtain up to $\lfloor \beta\, n \rfloor$ elements. In addition, it is easy to see that if $(N < N'$ and $m \leq m')$ then $L(N, D, m) \leq L(N', D, m')$. As a consequence, we get

$$L(N, \rho\, n, m_{\min}) \leq L(\lfloor \beta\, n \rfloor, \rho\, n, \widetilde{m}).$$

From this observation, we get the following theorem.

Theorem VI.12: Given an accurate pair of rates (α, β) for a flow of n elements, any receiver can use PolyReconstruct which outputs a list having at most $L(\lfloor \beta\, n \rfloor, \rho\, n, \widetilde{m})$ elements.

The previous theorem is also valid for the schemes from [14], [17]–[21]. We will now make a practical comparison between $L(\lfloor \beta\, n \rfloor, \rho\, n, \widetilde{m})$ and the bound $U(n)$ proposed in [18]. Our comparison will focus on the case $n = 1000$ as this is the value chosen in [18], [29] to illustrate the bound $U(n)$. Table III summarizes this comparison where the values from $U(1000)$ have been taken from [29]. The parameter r has been chosen as representing 10%, 30%, 50%, 70% and 90%.

The comparison seems to imply that the bound $L(\lfloor \beta\, n \rfloor, \rho\, n, \widetilde{m})$ is tighter than $U(n)$. Moreover, it seems that $L(\lfloor \beta\, n \rfloor, \rho\, n, \widetilde{m})$ achieves its smallest value for $r = \frac{1}{2}$ (i.e., $\rho = \frac{\alpha^2}{2\, \beta}$) which was already observed for $U(n)$ in [18]. As remarked in Section VI-C-1, in such a case, we can use the Sudan algorithm since $\widetilde{m} = 1$.

3) Application to Column Deletion: We now combine the results from Sections VI-C-1 and VI-C-2 to use them in the context of column deletion. We obtain the following properties the proofs of which are straightforward.

Corollary VI.13: Let ℓ be any integer no smaller than $\ell_{\min}$. The RS code from Section VI-A is

TABLE III
COMPARISON BETWEEN $U(n)$ (LEFT) AND $L(\lfloor \beta\, n \rfloor, \rho\, n, \widetilde{m})$ (RIGHT) FOR $n = 1000$

	(α,β)			
	$(0.5, 1.1)$	$(0.5, 1.25)$	$(0.5, 1.5)$	$(0.5, 2)$
10%	48|9	55|10	66|12	88|17
30%	20|5	23|5	28|7	38|9
50%	17|3	19|4	23|5	31|7
70%	20|8	23|9	28|11	38|16
90%	48|21	55|24	66|29	88|39
	(α,β)			
	$(0.75, 1.1)$	$(0.75, 1.25)$	$(0.75, 1.5)$	$(0.75, 2)$
10%	21|6	24|6	29|8	39|11
30%	9|3	10|3	12|4	16|6
50%	7|2	8|2	10|3	14|4
70%	9|5	10|6	12|7	16|10
90%	21|14	24|16	28|19	36|26
	(α,β)			
	$(0.8, 1.1)$	$(0.8, 1.25)$	$(0.8, 1.5)$	$(0.8, 2)$
10%	19|5	21|6	26|7	34|10
30%	8|3	9|3	11|4	14|5
50%	6|2	7|2	9|3	12|4
70%	8|5	9|5	10|7	14|9
90%	19|13	21|15	26|18	34|24
	(α,β)			
	$(0.9, 1.1)$	$(0.9, 1.25)$	$(0.9, 1.5)$	$(0.9, 2)$
10%	15|4	17|5	20|6	27|9
30%	6|2	7|3	8|3	11|5
50%	5|1	6|2	7|2	9|3
70%	6|4	7|5	8|6	11|8
90%	15|11	17|13	20|16	27|21

$\left(\frac{\lceil \alpha\, n \rceil - d}{n}, \ell, L(\lfloor \beta\, n \rfloor - d\,(\ell+1), \rho\, n, \widetilde{m}_{d,n})\right)$-list recoverable from any set of at most $\lfloor \beta\, n \rfloor - d\,(\ell+1)$ points where d is any integer in $\{0,\ldots,\mathcal{D}_\ell\}$ using $\widetilde{m}_{d,n}$ as interpolation multiplicity for PolyReconstruct where

$$\widetilde{m}_{d,n} := \left\lceil \frac{1}{\frac{\alpha^2}{\rho\left(\beta - \frac{d\,(\ell+1)}{n}\right)} - 1} \right\rceil .$$

Corollary VI.14: Let ℓ be any integer no smaller than $\ell_{\min}$. We have

$$\forall d \in \{0,\ldots,\mathcal{D}_\ell\} \quad \begin{cases} \widetilde{m}_{d,n} \le \widetilde{m} \\ L(\lfloor \beta\, n \rfloor - d\,(\ell+1), \rho\, n, \widetilde{m}_{d,n}) \le L(\lfloor \beta\, n \rfloor, \rho\, n, \widetilde{m}). \end{cases}$$

4) Practical Efficiency: As discussed in Section VI-C-4, two codes are utilized in our authentication protocol and the code sustaining most of the computational cost is the one used for list decoding. In the section, we used an RS code for this purpose.

Due to (3), the packet overhead is $\Omega(\log_2 n)$ bits ***but*** it is independent of the packet length $\mathcal{P}$. It should be noticed that the field $\mathbb{F}_{2^q}$ used for our RS code is the same as in [14]. In the recent full version of that paper [15], Lysyanskaya *et al.* provided some indicative values for q (see Table I from [15]). In those many examples, they approximated q by the ratio $\frac{\mathcal{H}}{\rho}$ and they used $\mathcal{H} = 160$ bits (SHA-1 hashing algorithm). The smallest value for q presented in [15] is 472 meaning that the field has 2^{472} elements. In their analysis, they indicated that the field operations required by PolyReconstruct may or may not be faster than the sign-each approach depending on the choices of parameters $\mathcal{H}$, $\mathcal{S}$, ρ and n. This drawback (also experienced by our scheme) comes from the implementation efficiency of the list decoding algorithm PolyReconstruct.

Lots of research have been done about PolyReconstruct since its creation by Guruswami and Sudan in 1999. It appears that the interpolation step is computationally expensive even if, in the complexity point of view, it is polynomial in the code parameters. That is why recent works on that topic have been dedicated to reduce the cost of this phase. In [51], Trifonov constructed an algorithm finding Groebner bases for polynomial ideal multiplication more efficiently. Its running time was later improved in [52] using a simplified formula for polynomial ideal squaring and a modification of the Karatsuba algorithm for triangular bivariate polynomials. In [53], Chen *et al.* refined the way polynomials were eliminated within the Kötter algorithm [54] used to perform interpolation. Though, these papers present techniques allowing to reduce the complexity of PolyReconstruct, the implementations done by the respective authors are only performed over very small fields ($\mathbb{F}_{2^5}$, $\mathbb{F}_{2^6}$ and $\mathbb{F}_{2^8}$ in [51]; $\mathbb{F}_{2^5}$ and $\mathbb{F}_{2^8}$ in [52]; $\mathbb{F}_{2^6}$ in [53]) or on specific communication channels (additive white Gaussian noise channel and Rayleigh fading channel in [53]).

Given the field parameters required for our authentication protocol (which are identical to [15]), it is likely that this technique be still expensive (in a practical point of view) based on the community's current knowledge about the implementation efficiency of PolyReconstruct.

D. Comparison to Lysyanskaya et al.'s Work

As said in Section I, [14] was the first paper using PolyReconstruct as a subroutine to achieve streaming authentication and its full version was recently published in [15]. Because of its similarities with our work, we are to argue about the fundamental differences between [15] and our current paper.

In [15], the authors exclusively focussed on RS codes. At the end of their paper, Lysyanskaya *et al.* formulated three open problems:

O1 Can the decoding procedure be simplified in order to reduce its time complexity?

O2 How efficient can the practical implementation of [15] be?

O3 Can other classes of error correcting codes provide better runtime or overhead?

In our paper, we addressed those three problems.

O1 We first used the truncation technique to speed-up the signature verification procedure which plays a central role in the authentication scheme. When our scheme is instantiated with a RS code (Section VI) like [15], we performed an analysis of the interpolation multiplicity of PolyReconstruct and we showed how receivers could appropriately tune it. To the best of our knowledge, this is the first time this efficiency parameter has been analyzed in the context of stream authentication.

O2 In Section VI-C-4, we analyzed the practical efficiency of PolyReconstruct. Based on the current knowledge of the scientific community, we argued that solutions based on PolyReconstruct were still rather theoretical than practical given the size of the finite field on which PolyReconstruct is supposed to operate to provide data authentication.

O3 In Section IV, we showed how ***any*** list decodable code could be used to authenticate digital streams as soon as their parameters hold (1). We also provided a detailed study of the packet overhead for this new class of protocols in Section V. Our work generalizes [14] to a broader family of authentication protocols in a black box way since no assumption about the list recoverable code is made except (1).

Apart from these considerations, our paper also makes use of two coding techniques. The first one is for data authentication via a list decodable code as in [15] where this role was played by a RS code. Contrary to Lysyanskaya *et al.*'s paper, we use a second code to correct data erasures. This allows any receiver to reconstruct the whole data stream which is a feature that [15] does not have. Furthermore,, we also correct a claim about the packet overhead stated in [15]. Indeed, at the end of Section I and in Section VI of their paper, Lysyanskaya *et al.* asserted that the packet overhead for their scheme was independant of n. We showed that their communication cost analysis done in Section V of [15] was incorrect since the use of a RS code implied that the underlying field $\mathbb{F}_{2^{\tilde{q}}}$ had to have at least n distinct values thus creating a $\Omega(\log_2 n)$-bit packet overhead (see (3) in Section VI-B).

VII. Concluding Discussion

In this paper, we presented an authentication protocol based on list recoverable codes for stream distribution over adversarial networks. Our construction is provably secure. As [14], [18], [21], our protocol enables total recovery of the data stream and it allows new participants to join the communication group at any block boundary. Our scheme works as a black-box construction since any list recoverable code whose parameters satisfy (1) can be used. As a consequence, one only has to focus on the computational complexity of the underlying list recoverable code when implementing our protocol.

We illustrated this construction with RS codes and we demonstrated that the RS code-based approaches developed in [14], [18], [21] could be regarded as particular cases of our scheme. We showed that our deletion process led to better bounds on the number of signature verification queries than in [14], [18], [21] while having the same packet overhead and computational complexity. Finally, we performed a study of the interpolation multiplicity to be used for PolyReconstruct. Based on McEliece's survey [34], we obtained an explicit bound for this parameter and deduced a new upper bound on the size of the list output by this reconstruction algorithm. That bound is also valid for the previous constructions using PolyReconstruct as a subroutine such as [14], [17]–[21].

As discussed in Section VI-C-4, our current knowledge about PolyReconstruct may still prevent us from having efficient implementations. However, our protocol is not the first cryptographic primitive based on the PRP. The first paper treating polynomial reconstruction in a cryptographic context dates from 1999 [55]. In that paper, Naor and Pinkas looked at the noisy PRP as a hardness assumption to design protocols for oblivious polynomial evaluation (OPE) having applications for password authentication and e-commerce (see [56] for extra details and applications). In [57], the hardness of solving the noisy PRP was used to provide semantic security for OPE and to construct a pseudorandom extender as well as a semantically secure cipher with forward security, superpolynomial message length and an error-correcting decryption algorithm. In [49], Augot and Finiasz constructed a public-key cryptosystem whose security is related to the PRP (even if no formal reduction was proposed). As explained in [58], it is fundamental to develop provable security frameworks not based on either the factoring or the discrete logarithm problems. All these papers are related to the post-quatum cryptography area [59] since the PRP has been shown to be NP-hard [60]. However, our research is in a slightly different context. Indeed, like Lysyanskaya *et al.*'s scheme ([14], [15]), our construction needs the PRP to be solvable in polynomial time since PolyReconstruct is to be used. Thus, our work should rather be classified in the same category as [61] (cryptanalysis of an optimized version of [49]) and [62] (identifying extra traitors beyond the security level of traitor tracing scheme). Therefore, the results of those later papers suffer from the same lack of practical efficiency of PolyReconstruct.

Based on these considerations, several research directions can be derived. First, one may want to consider other algorithms to list decode RS codes. Note that, to fit our authentication purpose, such an algorithm has to handle the fact that some x_i's of the PRP input can repeat themselves (pollution attacks) which is a property achieved by PolyReconstruct ([35], [47]). Second, our general scheme is secure for any list recoverable code satisfying (1). Would there exist a code having a practically more efficient list decoding algorithm? Third, in a more general aspect, it would be worth studying alternate approaches to list recoverable codes and cryptographic accumulators in order to handle adversarial injections of data as this is the central issue for stream authentication. Indeed, existing provably secure techniques are based on either techniques and we saw throughout this paper that none of these approaches is yet able to provide provable security, small overhead as well as both theoretical and practical efficient computations.

Appendix I
Proof of Theorem V.2

Assume that the scheme is either insecure or not (α, β)-correct. By definition, a PPT opponent $\mathcal{O}$ can break the scheme security or correctness with a non-negligible probability $\pi(\chi)$ where χ is the security parameter setting up both the digital signature and the hash function. Therefore, we must have either cases:

C1: With probability at least $\pi(\chi)/2$, $\mathcal{O}$ breaks the scheme correctness.

C2: With probability at least $\pi(\chi)/2$, $\mathcal{O}$ breaks the scheme security.

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

Case C1: We will demonstrate by contradiction 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 with non-negligible probability.

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 $\mathrm{Sign}_{\mathrm{SK}}$ (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 $(\mathrm{BID}_i, n_i, \alpha_i, \beta_i, \mathcal{P}_i, \mathrm{DP}_i)$ where DP_i is the set of n_i data packets of bit length $\mathcal{P}_i$ to be authenticated. In order to get the corresponding output, the signature is obtained by querying $\mathrm{Sign}_{\mathrm{SK}}$ as a black-box at Step 2 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 BID, $n, \alpha, \beta, \mathcal{P}$ and a set of received packets RP such that:

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

As said in Section VI-C-2, it is assumed that the list of polynomials contains only one polynomial $\mathcal{T}(X)$ per degree value so that field operations are uniquely determined by this list.

Assume that the digital signature is secure against chosen message attacks and the hash function is collision resistant.

Since $|\mathrm{RP} \cap \mathrm{AP}| \geq \lceil \alpha\, n \rceil$ and $|\mathrm{RP}| \leq \lfloor \beta\, n \rfloor$, Step 1 of Decoder ends successfully. Given the fact that the parameters of the list recoverable code satisfy (1), it is guaranteed that at least $\lambda\, n$ original $\widetilde{C}_i$'s are amongst the nonempty lists at the end of Step 3. Thus, the original codeword $(\widetilde{C}_1 \cdots \widetilde{C}_n)$ is in the list $\{\widetilde{\mathcal{C}}_1, \ldots, \widetilde{\mathcal{C}}_\mu\}$ output at Step 4.

It should be noticed that τ_{par} can be recovered from Table I. As a consequence, since the digital signature is unforgeable and the hash function is collision resistant, the pair message/signature going through the verification process at Step 5.2 correspond to DP. Therefore, at the end of Step 5, we have

$$\forall i \in \{1, \ldots, n\} \quad h_i = h(C_i).$$

For the same reason as before, at the end of Step 7, we have

$$\forall i \in \{1, \ldots, n\} \quad C'_i = \emptyset \quad \text{or} \quad C'_i = C_i.$$

Note that q and the pad length ℓ can also be computed from the values contained in Table I. Since $|\mathrm{RP} \cap \mathrm{AP}| \geq \lceil \alpha\, n \rceil$, the MDS decoder output the original message $(M_1 \cdots M_{\lceil \alpha\, n \rceil})$ during Step 8. Thus, at the end of Step 9, Decoder returns to the receiver the n original data packets, i.e.,

$$\forall i \in \{1, \ldots, n\} \quad P'_i = P_i.$$

We obtain a contradiction with our original hypothesis which stipulated

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

As a consequence, a forgery of the digital signature or a collision within the hash function occurred with non-negligible probability $\pi(\chi)/2$.

Case C2: We will demonstrate by contradiction 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 with non-negligible probability.

We consider the same kind of reduction as in Case C1. The opponent $\mathcal{O}$ breaks the security of the scheme if one of the following holds:

I. Authenticator was never queried on input BID, n, α, β, $\mathcal{P}$ and the decoding algorithm Decoder does not reject RP, i.e., $\{P'_1, \ldots, P'_n\} \neq \emptyset$ where $\{P'_1, \ldots, P'_n\} = \mathrm{Decoder}(\mathrm{PK}, \mathrm{BID}, n, \alpha, \beta, \mathcal{P}, \mathrm{RP})$.

II. Authenticator was queried on input BID, n, α, β, $\mathcal{P}$ for some data packets $\mathrm{DP} = \{P_1, \ldots, P_n\}$. Nevertheless, the output of Decoder verifies $P'_j \neq P_j$ for some $j \in \{1, \ldots, n\}$.

Sub-Case C2-I: Since Decoder output some nonempty packets, Step 5 had to terminate successfully. Thus, it has been found a pair $(h(\mathrm{BID} \| \tau_{\mathrm{par}} \| h_1 \| \cdots \| h_n), \sigma)$ such that

$$\mathrm{Verify}_{\mathrm{PK}}(h(\mathrm{BID} \| \tau_{\mathrm{par}} \| h_1 \| \cdots \| h_n), \sigma) = \mathrm{TRUE}.$$

If $\mathcal{O}$ never queried Authenticator for block tag BID then the previous pair is a forgery of the digital signature which was obtained with non negligible probability $\pi(\chi)/2$.

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

$$(\mathrm{BID}, \hat{n}, \hat{\alpha}, \hat{\beta}, \hat{\mathcal{P}}) \neq (\mathrm{BID}, n, \alpha, \rho, \mathcal{P}).$$

Therefore, the tag $\hat{\tau}_{\mathrm{par}}$ which was signed during $\mathcal{O}$'s query on block BID is different from the tag τ_{par} corresponding to the pair $(h(\mathrm{BID} \| \tau_{\mathrm{par}} \| h_1 \| \cdots \| h_n), \sigma)$. As a consequence, that pair is a forgery of the signature scheme which was obtained with non-negligible probability $\pi(\chi)/2$.

Sub-Case C2-II: We have the same situation as Case C1.

Appendix II
Proof of Theorem VI.3

It should be pointed out that the value of d a priori depends on each receiver and so does $U(n, \ell, d)$.

Denote N the number of points on which Poly-Reconstruct is run and T the number of original elements in this list. Since the pair of rates (α, β) is accurate, we have: $T \geq \lceil \alpha\, n \rceil$ and $N \leq \lfloor \beta\, n \rfloor$. As noticed before, we have: $T > \sqrt{(\rho\, n)\, N}$ which guarantees Poly-Reconstruct to be run successfully. Denote $\mathcal{L}(N, T, d)$ the size of the list output by Poly-Reconstruct. We want to prove: $\mathcal{L}(T, N, d) \leq U(n, \ell, d)$.

Using Theorem III.7 with $D = \rho\, n$, we have the following upper bound for $\mathcal{L}(T, N, d)$

$$\frac{T}{\rho\ n}\left(1 + \frac{(\rho\ n)\ N + \sqrt{(\rho\ n)^2\ N^2 + 4(T^2 - (\rho\ n)\ N)}}{2\ (T^2 - (\rho\ n)\ N)}\right) - \frac{1}{\rho\ n}. \tag{5}$$

First Bound: We have: $\forall (a, b) \in \mathbb{R}^+ \times \mathbb{R}^+$, $\sqrt{a + b} \leq \sqrt{a} + \sqrt{b}$. We deduce that $L(T, N, c)$ is upper bounded by

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

Because the receiver has deleted d columns, we have: $\lceil \alpha\, n \rceil - d \le T \le n - d$ and $\lceil \alpha\, n \rceil - d \le N \le \lfloor \beta\, n \rfloor - (\ell + 1)\, d$. We obtain the following inequalities:

$$\begin{aligned} T^2 - (\rho\, n)\, N &\ge (\lceil \alpha\, n \rceil - d)^2 - \rho\, n\, (\lfloor \beta\, n \rfloor - (\ell+1)\, d) \\ &\ge (\alpha\, n - d)^2 - \rho\, n\, (\beta\, n - (\ell+1)\, d). \end{aligned}$$

Due to our choice of ρ, we have: $(\alpha\, n - d)^2 - \rho\, n\, (\beta\, n - (\ell + 1)\, d) > 0$. As a consequence, $\mathcal{L}(T, N, d)$ is upper bounded by

$$\frac{n-d}{\rho\ n}\left(1+\frac{\rho\ n\ (\beta\ n-(\ell+1)\ d)}{(d-\alpha\ n)^2-\rho\ n\ (\beta\ n-(\ell+1)\ d)}+\frac{1}{\sqrt{(d-\alpha\ n)^2-\rho\ n\ (\beta\ n-(\ell+1)d)}}\right) - \frac{1}{\rho\ n}$$

which is equal to $U_1(n, \ell, d)$ after simplifying numerators and denominators by n^2. Since $\mathcal{L}(T, N, d)$ is an integer, we get: $\mathcal{L}(T, N, d) \le \lfloor U_1(n, \ell, d) \rfloor$.

Second Bound: We start again from (5). For similar reasons as above, it is easy to see that the numerator of the fraction is upper bounded by

$$\rho\ n^2\ \left[\left(\beta-\tfrac{\ell+1}{n}\ d\right)+\sqrt{\left(\beta-\tfrac{\ell+1}{n}\ d\right)^2+\tfrac{4}{\rho^2\ n^2}\ \left(\left(1-\tfrac{d}{n}\right)-\rho\left(\alpha-\tfrac{d}{n}\right)\right)}\right].$$

Using the same lower bound on $T^2 - (\rho\ n)\ N$ as before, we deduce: $\mathcal{L}(N, T, d) \le U_2(n, \ell, d)$ and thus $\mathcal{L}(T, N, d) \le \lfloor U_2(n, \ell, d) \rfloor$.

Finally: $\mathcal{L}(T, N, d) \le \min(\lfloor U_1(n, \ell, d) \rfloor, \lfloor U_2(n, \ell, d) \rfloor)$ which achieves the demonstration of our theorem.

Appendix III
Proof of Theorem VI.5

Let ℓ and d be as above. In order to prove $U(n, \ell, d) \le U(n)$, it is sufficient to demonstrate the following two inequalities:

$$U_1(n, \ell, d) \le U_1(n) \tag{6}$$
$$U_2(n, \ell, d) \le U_2(n). \tag{7}$$

Inequality (6): We compute the difference $\Delta_1(n, \ell, d) := U_1(n, \ell, d) - U_1(n)$. We have: $\Delta_1(n, \ell, d) = -\frac{d}{\rho\ n} + A(n, \ell, d) + \frac{1}{\rho\ n}\ B(n, \ell, d)$ where

$$\begin{cases} A(n, \ell, d) = \dfrac{\left(1 - \frac{d}{n}\right)\ \left(\beta - \frac{\ell+1}{n}\ d\right)}{\left(\alpha - \frac{d}{n}\right)^2 - \rho\ \left(\beta - \frac{\ell+1}{n}\ d\right)} - \dfrac{\beta}{\alpha^2 - \beta\,\rho} \\ B(n, \ell, d) = \dfrac{1 - \frac{d}{n}}{\sqrt{\left(\alpha - \frac{d}{n}\right)^2 - \rho\ \left(\beta - \frac{\ell+1}{n}\ d\right)}} - \dfrac{1}{\sqrt{\alpha^2 - \beta\,\rho}}. \end{cases}$$

We will demonstrate that $(A(n, \ell, d) \le 0)$ and $(B(n, \ell, d) \le 0)$ which will involve (6). Due to the definition of $\mathcal{D}_\ell$ and d, we get

$$d\,\frac{\ell+1}{n} < \frac{\lfloor \beta\, n \rfloor}{n} \le \beta.$$

Since $\ell \ge \left\lceil \frac{2\, \lfloor \beta\, n \rfloor}{\lceil \alpha\, n \rceil} \right\rceil - 1$, we deduce: $\frac{\lceil \alpha\, n \rceil}{2\, n} \ge \frac{\lfloor \beta\, n \rfloor}{n\ (\ell+1)}$. Therefore: $\frac{d}{n} < \frac{\lceil \alpha\, n \rceil}{2\, n} \le \frac{1}{2}$ which involves: $1 - \frac{d}{n} \in \left(\frac{1}{2}, 1\right]$. We obtain

$$0 < \left(1 - \frac{d}{n}\right)\ \left(\beta - \frac{\ell+1}{n}\ d\right) < \beta. \tag{8}$$

We also have

$$\left(\alpha - \frac{d}{n}\right)^2 - \rho\ \left(\beta - \frac{\ell+1}{n}\ d\right) - (\alpha^2 - \beta\,\rho) = \frac{d}{n}\ \left(\frac{d}{n} + \ell - 1\right)$$

Since $\ell \ge 1$, we deduce

$$\left(\alpha - \frac{d}{n}\right)^2 - \rho\ \left(\beta - \frac{\ell+1}{n}\ d\right) \ge \alpha^2 - \beta\,\rho > 0. \tag{9}$$

Combining Inequalities (8) and (9), we obtain: $A(n, \ell, d) \le 0$.

As the mapping $x \mapsto \sqrt{x}$ is strictly increasing over $\mathbb{R}^+$, we get

$$\sqrt{\left(\alpha - \frac{d}{n}\right)^2 - \rho\ \left(\beta - \frac{\ell+1}{n}\ d\right)} \ge \sqrt{\alpha^2 - \beta\,\rho} > 0. \tag{10}$$

Combining the fact that $1 - \frac{d}{n} \in \left(\frac{1}{2}, 1\right]$ with (10), we obtain: $B(n, \ell, d) \le 0$. Therefore, we deduce (6).

Inequality (7): As before, we compute the difference $\Delta_2(n, \ell, d) := U_2(n, \ell, d) - U_2(n)$. We have: $\Delta_2(n, \ell, d) = -\frac{d}{\rho\ n} + \frac{1}{2}\ C(n, \ell, d)$ where $C(n, \ell, d)$ is defined (see the equation at the bottom of the page).

Inequality (9) gives us a relation for the denominators of $C(n, \ell, d)$. As $(1 - \frac{d}{n}) \le 1$, it is sufficient to demonstrate

$$\begin{gathered} \left(\beta-\tfrac{\ell+1}{n}\ d\right)+\sqrt{\left(\beta-\tfrac{\ell+1}{n}\ d\right)^2+\tfrac{4}{\rho^2\ n^2}\ \left(\left(1-\tfrac{d}{n}\right)^2-\rho\ \left(\alpha-\tfrac{d}{n}\right)\right)} \\ \le \\ \beta + \sqrt{\beta^2 + \tfrac{4}{\rho^2\ n^2}\ (1 - \rho\alpha)} \end{gathered} \tag{11}$$

in order to obtain $C(n, \ell, d) \le 0$. It is easy to see that (11) is verified as soon as

$$\left[\left(\beta-\tfrac{\ell+1}{n}\ d\right)^2-\beta^2\right]+\tfrac{4}{\rho^2\ n^2}\ \left[\left(\left(1-\tfrac{d}{n}\right)^2-1\right)-\rho\ \tfrac{d}{n}\right]\le 0.$$

Due to the definitions of d and ℓ, we have

$$0 \le \beta - \frac{\ell+1}{n}\ d \le \beta \qquad \text{and} \qquad 0 \le 1 - \frac{d}{n} \le 1.$$

Therefore

$$\left(\beta - \frac{\ell+1}{n}\ d\right)^2 - \beta^2 \le 0 \qquad \text{and} \qquad \left(1 - \frac{d}{n}\right)^2 - 1 \le 0.$$

This involves that (11) is verified. We deduce that (7) holds which achieves the proof of our theorem.

$$C(n, \ell, d) = \left(1 - \frac{d}{n}\right)\ \frac{\left(\beta - \frac{\ell+1}{n}\ d\right) + \sqrt{\left(\beta - \frac{\ell+1}{n}\ d\right)^2 + \frac{4}{\rho^2\ n^2}\ \left(\left(1 - \frac{d}{n}\right)^2 - \rho\ \left(\alpha - \frac{d}{n}\right)\right)}}{\left(\alpha - \frac{d}{n}\right)^2 - \rho\ \left(\beta - \frac{\ell+1}{n}\ d\right)} - \frac{\beta + \sqrt{\beta^2 + \frac{4}{\rho^2\ n^2}\ (1 - \rho\alpha)}}{\alpha^2 - \beta\,\rho}$$

Acknowledgment

The authors would like to thank the reviewers for their suggestions to improve the quality of this paper.

References

[1] J. Postel, User Datagram Protocol, Internet standard RFC 0768, 1980 [Online]. Available: http://www.ietf.org/rfc/rfc0768.txt

[2] Y. Challal, H. Bettahar, and A. Bouabdallah, "A taxonomy of multicast data origin authentication: Issues and solutions," *IEEE Commun. Surveys Tuts.*, vol. 6, no. 3, pp. 34–57, Oct. 2004.

[3] A. Perrig, R. Canetti, J. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," in *Proc. 2000 IEEE Symp. Security and Privacy.*, 2000, pp. 56–73.

[4] A. Perrig and J. D. Tygar, *Secure Broadcast Communication in Wired and Wireless Networks*. Boston, MA: Kluwer, 2003.

[5] P. Golle and N. Modadugu, "Authenticating streamed data in the presence of random packet loss," in *Proc. 8th Annu. Symp. Network and Distributed System Security.*, 2001, pp. 13–22.

[6] S. Miner and J. Staddon, "Graph-based authentication of digital streams," in *Proc. 2001 IEEE Symp. Security and Privacy.*, 2001, pp. 232–246.

[7] V. Paxson, "End-to-end Internet packet dynamics," *IEEE/ACM Trans. Netw.*, vol. 7, no. 3, pp. 277–292, Jun. 1999.

[8] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modeling of the temporal dependence in packet loss," in *Proc. 1999 IEEE Conf. Computer Communications*, 1999, vol. 1, pp. 345–352.

[9] M. Al-Ibrahim and J. Pieprzyk, "Authenticating multicast streams in lossy channels using threshold techniques," in *Proc. 1st Int. Conf. Networking*, 2001, vol. 2094, pp. 239–249.

[10] Y. Desmedt and G. Jakimoski, "Non-degrading erasure-tolerant information authentication with an application to multicast stream authentication over lossy channels," in *Proc. Topics in Cryptology—CT-RSA 2007*, 2007, vol. 4377, pp. 324–338.

[11] A. Pannetrat and R. Molva, "Authenticating real time packet streams and multicasts," presented at the 7th Int. Symp. Computers and Communications., 2002.

[12] J. M. Park, E. K. P. Chong, and H. J. Siegel, "Efficient multicast packet authentication using signature amortization," in *Proc. 2002 IEEE Symp. Security and Privacy.*, 2002, pp. 227–240.

[13] Y. Park and Y. Cho, "The eSAIDA stream authentication scheme," *Comput. Sci. Appl.*, vol. 3046, pp. 799–807, 2004.

[14] A. Lysyanskaya, R. Tamassia, and N. Triandopoulos, "Multicast authentication in fully adversarial networks," in *Proc. 2003 IEEE Symp. Security and Privacy.*, 2003, pp. 241–253.

[15] A. Lysyanskaya, R. Tamassia, and N. Triandopoulos, "Authenticated error-correcting codes with applications to multicast authentication," *ACM Trans. Inf. Syst. Security*, vol. 13, no. 2, pp. 1–34, Feb. 2010.

[16] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.

[17] C. Tartary and H. Wang, "Efficient multicast stream authentication for the fully adversarial network," in *Proc. 6th Int. Workshop on Information Security Applications*, 2005, vol. 3786, pp. 108–125.

[18] C. Tartary and H. Wang, "Achieving multicast stream authentication using MDS codes," in *Proc. 5th Int. Conf. Cryptology and Network Security*, 2006, vol. 4301, pp. 108–125.

[19] C. Tartary and H. Wang, "Rateless codes for the multicast stream authentication problem," in *Proc. 1st Int. Workshop on Security*, 2006, vol. 4266, pp. 136–151.

[20] C. Tartary and H. Wang, "Combining prediction hashing and MDS codes for efficient multicast stream authentication," in *Proc. 12th Australasian Conf. Information Security and Privacy*, 2007, vol. 4586, pp. 293–307.

[21] C. Tartary, H. Wang, and J. Pieprzyk, "An hybrid approach for efficient multicast stream authentication over unsecured channels," in *Proc. 1st Int. Conf. Provable Security*, 2007, vol. 4784, pp. 17–34.

[22] C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. D. Tygar, "Distillation codes and applications to dos resistant multicast authentication," presented at the 11th Network and Distributed Systems Security Symp., 2004.

[23] J. Benaloh and M. de Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Proc. Advances in Cryptology—Eurocrypt '93*, 1993, vol. 765, pp. 274–285.

[24] R. Merkle, "A certified digital signature," in *Proc. Advances in Cryptology—Crypto'89*, 1989, vol. 435, pp. 218–238.

[25] R. Di Pietro, S. Chessa, and P. Maestrini, "Computation memory and bandwidth efficient distillation codes to mitigate DoS in multicast," in *Proc. 1st Int. Conf. Security and Privacy for Emerging Areas in Communication Networks.*, 2005, pp. 13–22.

[26] J. He, G. Xu, X. Fu, and Z. Zhou, "A hybrid and efficient scheme of multicast source authentication," in *Proc. 8th ACIS Int. Conf. Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, 2007, vol. 2, pp. 123–125.

[27] C. K. Wong and S. S. Lam, "Digital signatures for flows and multicasts," *IEEE/ACM Trans. Netw.*, vol. 7, no. 4, pp. 502–513, Aug. 1999.

[28] Y. Zhou and Y. Fang, "Multimedia broadcast authentication based on batch signature," *IEEE Commun. Mag.*, vol. 45, no. 8, pp. 72–77, Aug. 2007.

[29] C. Tartary, "Authentication for Multicast Communication," Ph.D. dissertation, Dept. Comput., Macquarie Univ., NSW, Australia, 2007.

[30] C. Tartary and H. Wang, "Efficient multicast stream authentication for the fully adversarial network," *Int. J. Security and Network*, vol. 2, no. 3/4, pp. 175–191, 2007.

[31] L. Nguyen, "Accumulators from bilinear pairings and applications," in *Proc. Topics in Cryptology CT-RSA 2005*, 2005, vol. 3376, pp. 275–292.

[32] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, ser. Discrete Mathematics and Its Applications. Boca Raton, FL: Chapman & Hall/CRC, 2006.

[33] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *J. Crypt.*, vol. 17, no. 4, pp. 297–319, Sep. 2004.

[34] R. J. McEliece, The Guruswami-Sudan Decoding Algorithm for Reed–Solomon Codes, NASA—Jet Propulsion Laboratory, 2003, Tech. Rep. IPN Progress Report 42-153.

[35] V. Guruswami and M. Sudan, "Improved decoding of Reed–Solomon and algebraic-geometric codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 1757–1767, Sep. 1999.

[36] O. Goldreich, *Foundations of Cryptography: Volume I—Basic Tools*. Cambridge, U.K.: Cambridge Univ. Press, 2001.

[37] D. R. Stinson, *Cryptography: Theory and Practice*, ser. Discrete Mathematics and Its Applications, 3rd ed. Boca Raton, FL: Chapman & Hall/CRC, 2006.

[38] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC, 1996.

[39] J.-P. Zanotti, Le Code Correcteur C.I.R.C [Online]. Available: http://zanotti.univ-tln.fr/enseignement/divers/chapter3.html

[40] J. Lacan and J. Fimes, "Systematic MDS erasure codes based on Vandermonde matrices," *IEEE Commun. Lett.*, vol. 8, no. 9, pp. 570–572, Sep. 2004.

[41] P. Elias, "List decoding for noisy channels," in *Proc. IRE Wescon Conf. Record (Part2)*, 1957, pp. 94–104.

[42] J. M. Wozencraft, "List decoding," in *Quarterly Progr. Rep.,* Res. Lab. Electron., Massachusetts Inst. Technol., 1958, vol. 48, pp. 90–95.

[43] V. Guruswami, *Algorithmic Results in List Decoding*, ser. Foundations and Trends® in Theoretical Computer Science. Boston, MA: Now Publishers, 2006, vol. 2.

[44] V. Guruswami, *List Decoding of Error-Correcting Codes*. New York: Springer-Verlag, 2004.

[45] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Hoboken, NJ: Wiley Interscience, 2005.

[46] H. O'Keeffe and P. Fitzpatrick, "Gröbner basis solutions of constrained interpolation problems," *Linear Algebra Appl.*, vol. 351–352, pp. 533–551, Aug. 2002.

[47] D. Bleichenbacher and P. Q. Nguyen, "Noisy polynomial interpolation and noisy Chinese remaindering," in *Proc. Advances in Cryptology—Eurocrypt'00*, May 2000, vol. 1807, pp. 53–69.

[48] D. Dolev and A. C.-C. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. 29, no. 2, pp. 198–208, Mar. 1983.

[49] D. Augot and M. Finiasz, "A public key encryption scheme based on the polynomial reconstruction problem," in *Proc. Advances in Cryptology—Eurocrypt '03*, 2003, vol. 2656, pp. 229–240.

[50] M. Sudan, "Decoding of Reed–Solomon codes beyond the error-correction bound," *J. Complexity*, vol. 13, no. 1, pp. 180–193, Mar. 1997.

[51] P. V. Trifonov, "Efficient interpolation in the Guruswami-Sudan algorithm," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4341–4349, Sep. 2010.

[52] P. Trifonov, "Implementing the interpolation step in the Guruswami-Sudan algorithm," in *Proc. XII Int. Symp. Problems of Redundancy in Information and Control Systems*, May 2009, pp. 109–113.

[53] L. Chen, R. A. Carrasco, and E. G. Chester, "Performance of Reed–Solomon codes using the Guruswami-Sudan algorithm with improved interpolation efficiency," *IET Commun.*, vol. 1, no. 2, pp. 241–250, Apr. 2007.

[54] R. Kötter, "Fast generalized minimum-distance decoding of algebraic-geometric and Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 3, pp. 721–736, May 1996.

[55] M. Naor and B. Pinkas, "Oblivious transfer and polynomial evaluation," in *Proc. 31st Annu. ACM Symp. Theory of Computing*, May 1999, pp. 245–254.

[56] M. Naor and B. Pinkas, "Oblivious polynomial evaluation," *SIAM J. Comput.*, vol. 35, no. 5, pp. 1254–1281, 2006.

[57] A. Kiayias and M. Yung, "Cryptographic hardness based on the decoding of Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 54, no. 6, pp. 2752–2769, Jun. 2008.

[58] A. Kiayias and M. Yung, "Directions in polynomial reconstruction based cryptography," *IEICE Trans.*, vol. E87-A, no. 5, pp. 978–985, May 2004.

[59] , D. J. Bernstein, J. Buchmann, and E. Dahmen, Eds., *Post-Quantum Cryptography*. New York: Springer, 2009.

[60] O. Goldreich, R. Rubinfeld, and M. Sudan, "Learning polynomials with queries: The highly noisy case," *SIAM J. Discrete Math.*, vol. 13, no. 4, pp. 535–570, 2000.

[61] A. Kiayias and M. Yung, "Cryptanalyzing the polynomial-reconstruction based public-key system under optimal parameter choice," *Des. Codes Cryptogr.*, vol. 43, no. 2–3, pp. 61–78, Jun. 2007.

[62] P. Junod, A. Karlov, and A. K. Lenstra, "Improving the Boneh-Franklin trator tracing scheme," in *Proc. 12th Int. Workshop on Practice and Theory in Public Key Cryptography*, Mar. 2009, vol. 5443, pp. 88–104.

Christophe Tartary (M'05) received his Ph.D. in Computer Science from Macquarie University in 2008. He is currently an assistant professor at the Institute for Theoretical Computer Science within the Institute for Interdisciplinary Information Sciences at Tsinghua University (P. R. China). His research interests are in cryptography, information security and coding theory.

Huaxiong Wang obtained a Ph.D. in Mathematics from the University of Haifa, Israel (1996) and a Ph.D. in Computer Science from the University of Wollongong, Australia (2001). He is currently with Nanyang Technological University, Singapore. His research interests include cryptography, information security, coding theory, combinatorics and theoretical computer science. He is on the editorial boards of Designs, Codes and Cryptography, Journal of Communications and Journal of Communications and Networks and was the Program Co-Chair of 9th Australasian Conference on Information Security and Privacy (ACISP'04), Sydney, Australia, July, 2004 and 4th International Conference on Cryptology and Network Security (CANS05), Xiamen, Fujian, China, December, 2005. He won the inaugural Prize of Research Contribution awarded by the Computer Science Association of Australasia in 2004.

San Ling received the B.A. degree in mathematics from the University of Cambridge, Cambridge, U.K., in 1985, and the Ph.D. degree in mathematics from the University of California, Berkeley, in 1990. Since April 2005, he has been a Professor with the Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. Prior to that, he was with the Department of Mathematics, National University of Singapore. His research fields include arithmetic of modular curves and application of number theory to combinatorial designs, coding theory, cryptography, and sequences.