A New Class of Rateless Codes Based on Reed–Solomon Codes

Reza Rafie Borujeny, *Student Member, IEEE*, and Masoud Ardakani, *Senior Member, IEEE*

***Abstract*—Erasure codes, such as LT and Raptor codes, are designed for the purpose of erasure-resilient distribution of data over computer networks. To achieve a small reception overhead, however, LT and Raptor codes must be used with a large design length k, making these codes unsuitable for real-time applications. In this paper, we propose a new class of erasure codes based on Reed–Solomon codes that unlike other Reed–Solomon-based erasure codes are rateless and also, unlike other rateless codes, guarantee zero overhead even for small k. Moreover, they have a reasonable computational complexity of coding when k is not too large. In fact, a practical implementation of subfield subcodes of Reed-Solomon codes with arbitrarily large block lengths is presented.**

***Index Terms*—Rateless codes, Raptor codes, Reed-Solomon codes, Cauchy matrix, sub-field sub-code.**

I. Introduction

TRANSMISSION of a data packet over a network, such as the Internet, can accurately be modeled as data transmission over the erasure channel. Thus, designing efficient communication methods for the erasure channel is of great interest. One important case of data transmission is multicasting which is defined as transmission of one stream of data from one source to multiple destinations. Multicasting is used, for example, for Internet TV, teleconferencing or software update distributions. Therefore, multicasting over a group of erasure channels with possibly different erasure rates has been vastly studied.

A digital fountain approach is introduced in [1] as an efficient solution for multicasting over the erasure channel. LT codes [2] and Raptor codes [3] are then devised as practical fountain codes. To achieve a small reception overhead, however, LT and Raptor codes should be used with a large design length k [4]. Moreover, almost no decoding does happen before receiving k encoding packets. This late decoding can raise a concern when partial data recovery is needed, for example in real-time applications.

Reed-Solomon (RS) codes [5]–[7] are also investigated as approximations to a digital fountain [1]. As a class of maximum distance separable (MDS) codes, RS codes offer zero overhead even for small k. Moreover, systematic RS codes allow partial data recovery. RS codes, however, have a fixed block length. Thus, their *stretch factor* (defined as the ratio of the number of encoded packets to the number of information packets k) is fixed a priori which makes them not a true digital fountain. To address this issue, some authors suggested cycling the original encoding block in order to increase the stretch factor [8], [9]. The resulting code is no longer MDS, and may impose overhead on the receivers.

In this paper, we design a new class of fountain codes upon Cauchy-based RS codes. Our codes are rateless, i.e., they have an infinite stretch factor. Moreover, because of their MDS property, their reception overhead is zero for any design length k. Thus, for applications with small k, e.g. those that require partial data recovery, our rateless Reed-Solomon (RLRS) codes can be of interest.

In comparison with Random Linear Fountain codes [10], [11], RLRS codes guarantee zero overhead and a lower coding complexity due to their algebraic structure. In fact, we show that the coding complexity of our codes grow as $O(k^2)$ per block of data, while the complexity of Random Linear Fountain codes is known to grow as $O(k^3)$.

The main advantage of our RLRS codes over a general MDS code with a very large stretch factor is that our code has a lower coding complexity. More precisely, we later see that RLRS codes start with a low stretch factor (low complexity) and increase the stretch factor only if needed.

In comparison with Raptor codes that enjoy linear coding complexity, our RLRS codes are more complex. The worst case complexity of RLRS codes is determined by their kernel RS code. This suggests that RLRS codes can be of interest for applications with small block length (e.g., multimedia streaming). For such applications, the main benefit of RLRS codes over Raptor codes is that RLRS codes have zero overhead while short block length Raptor codes suffer from a relatively large overhead. It is worth mentioning that, non-binary Raptor codes (see e.g. [12]) are developed to outperform the conventional Raptor codes in terms of overhead. To have a better understanding of RLRS codes, in this paper, comparisons with non-binary Raptor codes are also provided (see Appendix II.)

The remainder of this paper is organized as follows. We first review Cauchy-based RS codes that play an important role in our design. In Section III, we introduce the idea of field extension for RS codes. Field extension will allow us to turn a fixed-length RS code to a truly rateless code. As discussed later, with a Cauchy-based RS code, field extension preserves the MDS property. In Section IV, we describe and study our RLRS

Manuscript received June 5, 2015; revised September 29, 2015; accepted November 13, 2015. Date of publication November 23, 2015; date of current version January 14, 2016. This work was supported by the TELUS Corporation and Natural Sciences and Engineering Research Council of Canada. The associate editor coordinating the review of this paper and approving it for publication was D. Declercq.

The authors are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada (e-mail: reza.rafie@ualberta.ca; ardakani@ualberta.ca).

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/TCOMM.2015.2502952

codes. We study the complexity of RLRS codes in Section V. Some discussions are provided in Section VI. Section VII concludes the paper.

II. Background: Cauchy-Based Reed-Solomon Codes

Given a design length k and a field size q, a Cauchy-based RS code over Galois field $\mathbb{F}_q, q > k$, encodes k packets of information into q encoded packets based on its $k \times q$ generator matrix G_q. The generator matrix of this Cauchy-based RS code is a systematic one given by

$$G_q = \left(g_{i,j}^{(q)}\right) = \left[I_k | P_q\right] \tag{1}$$

where I_k is the $k \times k$ identity matrix and $P_q = \left(p_{i,j}^{(q)}\right)$ is a $k \times (q-k)$ Cauchy matrix over $\mathbb{F}_q$ defined by

$$P_q = \left(p_{i,j}^{(q)}\right) = \begin{pmatrix} \frac{1}{x_1^{(q)}+y_1^{(q)}} & \frac{1}{x_1^{(q)}+y_2^{(q)}} & \cdots & \frac{1}{x_1^{(q)}+y_{q-k}^{(q)}} \\ \frac{1}{x_2^{(q)}+y_1^{(q)}} & \frac{1}{x_2^{(q)}+y_2^{(q)}} & \cdots & \frac{1}{x_2^{(q)}+y_{q-k}^{(q)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x_k^{(q)}+y_1^{(q)}} & \frac{1}{x_k^{(q)}+y_2^{(q)}} & \cdots & \frac{1}{x_k^{(q)}+y_{q-k}^{(q)}} \end{pmatrix} \tag{2}$$

where $x_i^{(q)}$ and $y_i^{(q)}$ are elements of $\mathbb{F}_q$ such that $x_i^{(q)} \neq x_j^{(q)}$, $y_i^{(q)} \neq y_j^{(q)}$ and $x_i^{(q)} \neq y_j^{(q)}$ for any $i \neq j$. Also, $x_i^{(q)} \neq y_i^{(q)}$ for all $i \leq min(k, q-k)$. Each square sub-matrix of a Cauchy matrix is a Cauchy matrix by itself and so is invertible [13].

Considering $q = 2^m$, we assume that a file of size $k \times m \times 2^B$ bits is to be transmitted from a source to a set of destinations. The file is chunked into k information packets of size $m \times 2^B$ bits $X_1, X_2, \ldots, X_k$ and each information packet is interpreted as a vector of 2^B symbols over $\mathbb{F}_q$. The jth output (encoded) packet from the source is then formed according to the jth column of G_q as follows

$$Y_j = \sum_{i=1}^{k} g_{i,j}^{(q)} X_i. \tag{3}$$

The MDS property of the code implies that every k columns of G_q form an invertible $k \times k$ matrix over $\mathbb{F}_q$. This means that the receiver is able to reconstruct the original file after receiving any k distinct encoded packets. Each received packet is a linear equation in k unknowns, i.e., the original k information packets. Although there are different approaches for decoding RS codes [13]–[19], we simply assume that the decoder performs matrix inversion to find the values of input packets. The complexity of matrix inversion for a general $k \times k$ system is $O(k^3)$. However, if the coefficient matrix of the system has a Cauchy form, matrix inversion can be performed with $O(k^2)$ complexity [20], [21].

The main problem of RS codes, when used as fountain codes, is that their stretch factor $\frac{q}{k}$ is fixed. This is because the block length q is fixed. Thus, RS codes can approximate a digital fountain only when their field size is very large. It can be seen that coding complexity grows with the field size. This shows that RS codes with large field size as digital fountains are not complexity efficient. Our RLRS codes, on the other hand, start with a small field size and increase it (i.e., increase complexity) only when needed.

III. Preservative Field Extension

In this section, we present the mathematical foundation needed for our code construction. For $q = 2^m$, each symbol in $\mathbb{F}_q$ can be represented as a binary-sequence of m bits. Field addition is nothing but bitwise **XOR** of the symbols. However, multiplication rules depend on the construction of the field. For constructing a field $\mathbb{F}_q$ with $q = 2^m$, we can first find a degree m irreducible polynomial $p(x)$ over $\mathbb{F}_2[x]$, i.e., over all polynomials with coefficients in $\mathbb{F}_2$. Such a polynomial always exists [22]. Then, we need to represent each field symbol by a polynomial of degree less than m. For instance, for a symbol $(b_{m-1}, b_{m-2}, \ldots, b_0)$ with $b_i \in \{0, 1\}$ the polynomial representation would be $\sum_{i=0}^{m-1} b_i x^i$. The multiplication rule for two symbols is multiplying their corresponding polynomials and finding the remainder after division by $p(x)$.

Remember that, when k is fixed, the stretch factor of RS codes is proportional to the field size in which they are built. One way to allow an unbounded stretch factor is to find a way to increase the field size q on-the-fly. To clarify this point, assume that an RS code is being encoded in $\mathbb{F}_q$. If transmission is not successful even after encoding q output packets, one may increase q to obtain a larger stretch factor. Changing q, however, may render all the previously received packets useless as they were formed in a different field size and with a different generator matrix.

A key step in our rateless coding solution is that we show that it is possible to increase q while keeping all the previously encoded/received packets valid and preserving the MDS property. This way, if with the original q and after q transmissions the reception is not completed, past encoded packets are re-interpreted as packets in the new field and encoding process continues by generating new output packets in the new field.

Definition 1. Consider a field $\mathbb{F}_q$ with $q = 2^m$ and another $\mathbb{F}_Q$ with $Q = 2^n$ for $n = 2m$. We say that $\mathbb{F}_Q$ is a preservative extension of $\mathbb{F}_q$ if $\mathbb{F}_Q$ keeps the binary-sequence representation of the symbols of $\mathbb{F}_q$ unchanged subject to a zero padding. More precisely, let us consider

$$a = (a_{m-1}, a_{m-2}, \ldots, a_0)_{1\times m} \in \mathbb{F}_q$$
$$b = (b_{m-1}, b_{m-2}, \ldots, b_0)_{1\times m} \in \mathbb{F}_q$$

with

$$a \times b = c = (c_{m-1}, c_{m-2}, \ldots, c_0)_{1\times m}$$

as their multiplication in $\mathbb{F}_q$. Now, for

$$a' = (0, 0, \ldots, 0, a_{m-1}, \ldots, a_0)_{1\times n} \in \mathbb{F}_Q$$
$$b' = (0, 0, \ldots, 0, b_{m-1}, \ldots, b_0)_{1\times n} \in \mathbb{F}_Q$$
$$b'' = (b_{m-1}, \ldots, b_0, 0, \ldots, 0)_{1\times n} \in \mathbb{F}_Q$$

their multiplication in $\mathbb{F}_Q$ must be given by

$$a' \times b' = (0, 0, \ldots, 0, c_{m-1}, \ldots, c_0)_{1\times n}$$
$$a' \times b'' = (c_{m-1}, \ldots, c_0, 0, 0, \ldots, 0)_{1\times n}. \quad (4)$$

Definition 1 implies that $\mathbb{F}_q$ is a sub-field (up to a zero padding) of its preservative extension. In the rest of this section, we investigate the existence of a preservative extension of a given field.

Lemma 1. For any field $\mathbb{F}_q$ with $q = 2^m$ there exists an irreducible polynomial $f(x) = hx^2 + x + 1$ for some $h \in \mathbb{F}_q$ and $h \neq 0$.

Proof: Equivalently, we show that there always exist a non-zero element $h \in \mathbb{F}_q$ such that the quadratic equation $hx^2 + x + 1$ has no roots in $\mathbb{F}_q$. This is also equivalent to showing that for at least one non-zero element h, $\mathrm{Tr}(h) \neq 0$ [23]. Here, $\mathrm{Tr}(x)$ is the trace of an element and is given by

$$\mathrm{Tr}(x) = x + x^2 + \cdots + x^{2^{m-1}}. \quad (5)$$

Note that $\mathrm{Tr}(x)$ is a polynomial of degree 2^{m-1} and has at most 2^{m-1} roots including zero in $\mathbb{F}_q$, but $q = 2^m$. Thus, we always can find a non-zero element $h \in \mathbb{F}_q$ such that $\mathrm{Tr}(h) \neq 0$, thus $f(x)$ is irreducible. A trivial value of h for an odd m is $h = 1$. ■

Theorem 1. For any $\mathbb{F}_q$ with $q = 2^m$ there exists a preservative field $\mathbb{F}_Q$ such that $Q = 2^{2m}$.

Proof: Consider an irreducible polynomial $f(x) = hx^2 + x + 1$ for some $h \in \mathbb{F}_q$ and $h \neq 0$ and construct $\mathbb{F}_Q$ with arithmetic modulo $f(x)$. In other words, using polynomial representation

$$\mathbb{F}_Q = \{\lambda x + \mu | \lambda \in \mathbb{F}_q, \mu \in \mathbb{F}_q\} \quad (6)$$

for two elements $\lambda_1 x + \mu_1$ and $\lambda_2 x + \mu_2$ in $\mathbb{F}_Q$, the field addition is defined as the polynomial summation and their multiplication is the remainder after division by $f(x)$. That is

$$(\lambda_1 x + \mu_1) \times (\lambda_2 x + \mu_2) = (\lambda_1\mu_2 + \lambda_2\mu_1 - \frac{\lambda_1\lambda_2}{h})x + \mu_1\mu_2 - \frac{\lambda_1\lambda_2}{h}. \quad (7)$$

To show that $\mathbb{F}_Q$ is a preservative extension of $\mathbb{F}_q$, we need to show that for any two members of $\mathbb{F}_q$ like a, b with $c = a \times b$ as their multiplication in $\mathbb{F}_q$, the multiplication $(0x + a) \times (0x + b)$ in $\mathbb{F}_Q$ is $(0x + c)$ and $(0x + a) \times (bx + 0)$ in $\mathbb{F}_Q$ is $cx + 0$ which is trivial according to the form of $f(x)$. ■

As a result of this theorem, the generator matrix of a Cauchy-based RS code keeps its Cauchy structure if we interpret the symbols in the extended field. In other words, the generator matrix of a Cauchy-based RS code in $\mathbb{F}_q$ is a sub-matrix of the generator matrix of a Cauchy-based RS code in $\mathbb{F}_Q$ when everything is interpreted in $\mathbb{F}_Q$. Moreover, the MDS property is preserved. We will discuss this property further in the next section.

IV. Digital Fountain With Reed-Solomon Codes

A. Communication Channel

Assume a memoryless packet erasure channel with erasure rate ϵ, i.e., each transmitted packet is received by the receiver correctly with probability $(1 - \epsilon)$ and is erased with probability ϵ. Also assume that there is an ideal feedback channel from the receiver to the transmitter. The receiver uses this channel to let the transmitter know whenever the received block of data is decodable.

B. Encoding and Decoding

Given a design length k and an initial field size q, the encoder at the transmitter side starts making the encoded packets based on the columns of the generator matrix G_q. It is assumed that each encoded packet carries an identifier that determines the corresponding column in the generator matrix. On the other side, the receiver listens to the channel until it receives exactly k encoded packets. At this point, the receiver asks the transmitter to stop transmitting by sending a feedback packet on the feedback channel. In case that the encoder reaches the qth encoded packet and does not hear any feedback from the receiver, it changes the field size to $Q = q^2$ and uses the generator matrix G_Q afterwards. The generator matrix G_Q is given by

$$G_Q = \left(g_{i,j}^{(Q)}\right) = \left[I_k | P_Q\right] \quad (8)$$

where I_k is the $k \times k$ identity matrix and $P_Q = \left(p_{i,j}^{(Q)}\right)$ is a $k \times (Q - k)$ Cauchy matrix over $\mathbb{F}_Q$ defined by

$$P_Q = \left(p_{i,j}^{(Q)}\right) = \begin{pmatrix} \frac{1}{x_1^{(Q)}+y_1^{(Q)}} & \frac{1}{x_1^{(Q)}+y_2^{(Q)}} & \cdots & \frac{1}{x_1^{(Q)}+y_{Q-k}^{(Q)}} \\ \frac{1}{x_2^{(Q)}+y_1^{(Q)}} & \frac{1}{x_2^{(Q)}+y_2^{(Q)}} & \cdots & \frac{1}{x_2^{(Q)}+y_{Q-k}^{(Q)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x_k^{(Q)}+y_1^{(Q)}} & \frac{1}{x_k^{(Q)}+y_2^{(Q)}} & \cdots & \frac{1}{x_k^{(Q)}+y_{Q-k}^{(Q)}} \end{pmatrix} \quad (9)$$

where $x_i^{(Q)}$ and $y_i^{(Q)}$ are elements of $\mathbb{F}_Q$ such that $x_i^{(Q)} \neq x_j^{(Q)}$, $y_i^{(Q)} \neq y_j^{(Q)}$ and $x_i^{(Q)} \neq y_j^{(Q)}$ for any $i \neq j$. Also, $x_i^{(Q)} \neq y_i^{(Q)}$ for all $i \leq k$. Moreover, $x_i^{(Q)}$ and $y_j^{(Q)}$ for $i \leq k$ and $j \leq q - k$ are the elements of $\mathbb{F}_q$ and the isomorphism that maps them to their equivalent elements in $\mathbb{F}_Q$ is simply a zero padding such that for $i \leq k$ and $j \leq q - k$

$$x_i^{(q)} \xrightarrow{zeropadding} x_i^{(Q)}$$
$$y_j^{(q)} \xrightarrow{zeropadding} y_j^{(Q)}. \quad (10)$$

In other words, G_Q is carefully designed over $\mathbb{F}_Q$, the preservative extension of $\mathbb{F}_q$, such that the first q columns are exactly (up to a zero padding at the prefix of their binary-sequence representation) the columns of G_q. Encoder continues with the $(q + 1)^{\text{th}}$ column of G_Q and goes forward waiting for the feedback message. Note that the packet length is not affected since each source packet is now treated as a vector of 2^{B-1} symbols from $\mathbb{F}_Q$. If the encoder reaches the Q^{th} encoded packet and still does not hear any feedback, it continues by using a preservative extension of $\mathbb{F}_Q$ and so forth. This means that we first use an initial stretch factor $\frac{q}{k}$ and if it is not enough, we increase it to $\frac{q^2}{k}$ and so on.

At the decoder, when k encoded packets are arrived, decoding becomes possible via solving a system of k linear equations. Even if field extensions are performed, the MDS property is preserved. This is because G_Q defines an MDS code and G_q is a sub-matrix of G_Q. A detailed example provided in Appendix I shows this fact. Moreover, this example shows that the packet size is not affected by field extension. As a conclusion, regardless of k, zero overhead is guaranteed by RLRS codes.

After receiving k encoded packets, our decoder first chooses its finite field size equal to the largest field size used by the transmitter, say s. Then, the decoder uses the received systematic packets and substitutes them in other equations. This reduces the size of the system of equations that should be solved by matrix inversion. Let us call the coefficient matrix of the remaining system of equations by $A_{\ell\times\ell}$. It is straightforward to see that A has a Cauchy form, thus its inverse can be found with $O(\ell^2)$ multiplications in $\mathbb{F}_s$. The inverse matrix A^{-1} is then used to recover the remaining information packets.

V. Complexity Analysis

As discussed earlier, our proposed RLRS code is rateless and offers zero reception overhead for all k. In this section, we study the coding complexity of this code and argue that with proper choice of parameters the average coding complexity can also be practical. By coding complexity we mean both the complexity of encoding and decoding measured as the number of field operations (additions and multiplications) per information packet.

Both encoding and decoding complexity depend on the number of field extensions performed, $e \geq 0$. This is because e determines the field size $s = 2^{m2^e}$ and therefore the complexity of field operations.

One can define a failure probability $P_f(e)$ for a given e as the probability that e field extensions are not enough for successful transmission of k encoded packets. We have

$$P_f(e) = \sum_{i=0}^{k-1} \binom{s}{i} \epsilon^{s-i}(1-\epsilon)^i \tag{11}$$

where $s = 2^{m\times 2^e}$ is the block length after e field extensions. Notice that s grows super-exponentially in e, meaning that e is very small in a practical setting. In fact, even $e = 1$ results in very low failure probabilities. For example, with $m = 8$ (i.e., initial field symbols are bytes), $k = 100$ and even $\epsilon = 0.99$ (far from any practical setup), we get $P_f(1) < 10^{-160}$. It is concluded that cases with $e \geq 2$ have no practical importance. The case $e = 1$ cannot be ignored because for example when $e = 0$ (no field extensions), $m = 8$, $k = 100$ and $\epsilon = 0.5$ we get $P_f(0) > 10^{-4}$ which is a non-negligible failure probability. As a result, in a practical scenario, we expect to see either no field extensions, or at most one field extension. Our complexity analysis is provided for a general $e \leq B$. In practice however, we only expect cases with $e < 2$.

A. Encoding Complexity

We average the encoding complexity over transmissions with at most B field extensions. To this aim, we should consider the cost of packet additions and symbol-by-packet multiplications. For a finite field $\mathbb{F}_q$ with $q = 2^m$, addition of two packets of length $m \times 2^B$ bits is performed by considering the packets as two vectors over $\mathbb{F}_q$ of length 2^B and adding the 2^B corresponding components of the vectors. Thus, if we denote the complexity of one symbol-by-symbol addition in $\mathbb{F}_q$ with $\mathcal{A}(m)$ each packet addition has $2^B \times \mathcal{A}(m)$ complexity. For multiplication of a field symbol by one packet in $\mathbb{F}_q$, we need to perform 2^B parallel symbol-by-symbol multiplications. The cost of one symbol-by-symbol multiplication in $\mathbb{F}_q$ is shown by $\mathcal{M}(m)$. In practice, we usually have $\mathcal{A}(m) = \eta\mathcal{M}(m)$ for some $\eta \leq 1$. The value of η highly depends on finite field arithmetic implementations. Considering the fact that encoding of one non-systematic packet requires k packet additions and k symbol-by-packet multiplications, the encoding complexity for encoding of one non-systematic encoded packet over $\mathbb{F}_q$ is

$$\chi(m) = \Big(2^B\mathcal{M}(m) + 2^B\mathcal{A}(m)\Big)k = 2^B(1+\eta)k\mathcal{M}(m). \tag{12}$$

After one field extension, each symbol in the extended field $\mathbb{F}_Q$ with $Q = 2^{2m}$ is represented by a binary sequence of length $2m$. The complexity of one symbol addition in $\mathbb{F}_Q$ is equivalent to complexity of two symbol additions in $\mathbb{F}_q$ with $q = 2^m$, i.e., $\mathcal{A}(2m) = 2\mathcal{A}(m)$. Each packet contains 2^{B-1} symbol in $\mathbb{F}_Q$. Thus, computational complexity of one packet addition over $\mathbb{F}_Q$ is $2^{B-1} \times \mathcal{A}(2m) = 2^B \times \mathcal{A}(m)$ which is the same as complexity of one packet addition over $\mathbb{F}_q$. The computational complexity of one symbol-by-symbol multiplication over $\mathbb{F}_Q$ can be found in terms of complexity of operations over $\mathbb{F}_q$. According to (7), each symbol-by-symbol multiplication in $\mathbb{F}_Q$ can be done by 4 multiplications and 3 additions in $\mathbb{F}_q$.[1] Thus, multiplication complexity in $\mathbb{F}_Q$ is

$$\mathcal{M}(2m) = 4\mathcal{M}(m) + 3\mathcal{A}(m) = (4+3\eta)\mathcal{M}(m). \tag{13}$$

Therefore, the encoding complexity of one non-systematic encoded packet over $\mathbb{F}_Q$ is

$$\begin{aligned}\chi(2m) &= \Big(2^{B-1}\mathcal{M}(2m) + 2^{B-1}\mathcal{A}(2m)\Big)k \\ &= 2^B\left(2 + \frac{5}{2}\eta\right)k\mathcal{M}(m).\end{aligned} \tag{14}$$

According to (12) and (14), encoding complexity of one non-systematic packet after one field extension grows by a factor of

$$\frac{\chi(2m)}{\chi(m)} = \frac{2+2.5\eta}{1+\eta} = 2 + 0.5\frac{\eta}{1+\eta} = \xi \tag{15}$$

where $\xi \leq 2.25$ represents the speed of complexity growth by field extension.

[1] Division by h is a symbol-by-symbol operation which is performed only once for each packet. Therefore, we neglect its computational complexity in our analysis.

Encoding complexity of RLRS codes depends on the number of packets encoded in each extended finite field. The average encoding complexity per information packet at the encoder, C_E, can be obtained by considering the average number of encoded packets over each extended field. We denote the average number of non-systematic encoded packets when no field extensions have performed by $\mathcal{K}_0$ which is

$$\mathcal{K}_0 = (1-\epsilon)^k \sum_{i=1}^{q-k} \binom{k+i-1}{k-1} i\epsilon^i + P_f(0)(q-k). \tag{16}$$

Similarly, denoted by $\mathcal{K}_e$, the average number of encoded packets after $e > 0$ field extension is

$$\mathcal{K}_e = (1-\epsilon)^k \sum_{i=1}^{q^{2^e}-q^{2^{e-1}}} \binom{q^{2^{e-1}}+i-1}{k-1} i\epsilon^{q^{2^{e-1}}+i-k} + P_f(e)\left(q^{2^e}-q^{2^{e-1}}\right). \tag{17}$$

The average encoding complexity per information packet then can be calculated as

$$C_E = \frac{1}{k}\sum_{e=0}^{B} \mathcal{K}_e \chi(2^e m) = \frac{\chi(m)}{k}\sum_{e=0}^{B} \mathcal{K}_e \xi^e \tag{18}$$

As we mentioned earlier, it is extremely unlikely that more than one field extensions are needed. Thus, assuming $B = 1$, (18) can be simplified to

$$C_E = \frac{\chi(m)}{k}\left(\mathcal{K}_0 + \left(2 + 0.5\frac{\eta}{1+\eta}\right)\mathcal{K}_1\right). \tag{19}$$

From (16), $\mathcal{K}_0$ can be upper-bounded by

$$\mathcal{K}_0 \le (1-\epsilon)^k \sum_{i=1}^{\infty} \binom{k+i-1}{k-1} i\epsilon^i \tag{20}$$

$$= (1-\epsilon)^k \epsilon \frac{\partial}{\partial \epsilon}\left(\epsilon^{-k}\sum_{i=1}^{\infty}\binom{k+i-1}{k-1}\epsilon^{k+i}\right) \tag{21}$$

$$= (1-\epsilon)^k \epsilon \frac{\partial}{\partial \epsilon}\left(\epsilon^{-k}\frac{\epsilon^{k-1}}{(1-\epsilon)^k} - 1\right) \tag{22}$$

$$= \frac{\epsilon k}{1-\epsilon}. \tag{23}$$

In derivation of (22) from (21), we used the fact that the generator polynomial of $\binom{m}{n}$ for $m = 0, 1, \ldots$ is $\frac{x^{n+1}}{(1-x)^n}$. With a similar argument we can show that

$$\mathcal{K}_1 \le P_f(0)\frac{\epsilon k}{1-\epsilon}. \tag{24}$$

Thus, the encoding complexity is upper-bounded by

$$C_E \le \chi(m)\frac{\epsilon}{1-\epsilon}\left(1 + \xi P_f(0)\right) \tag{25}$$

$$= 2^B(1+\eta)k\mathfrak{M}(m)\frac{\epsilon}{1-\epsilon}\left(1+\xi P_f(0)\right). \tag{26}$$

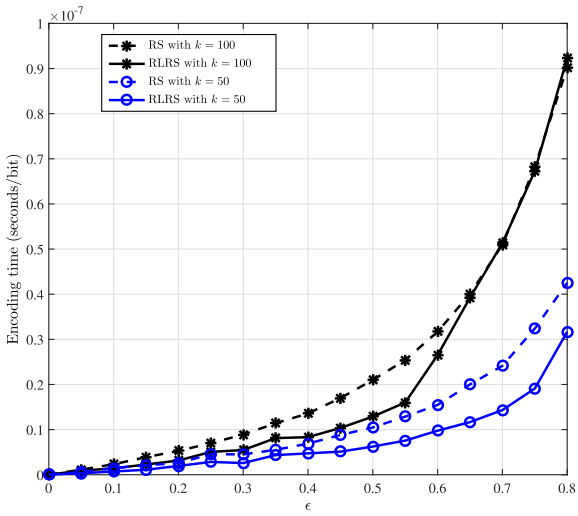

Fig. 1. Encoding time complexity per information bit for RLRS codes with initial field size $q = 256$ and Cauchy-based RS codes over a field of size $Q = q^2$. For $k = 50$, the effect of field extension becomes more pronounced when $\epsilon > 0.75$. The effect of field extension for $k = 100$, on the other hand, comes into play when $\epsilon > 0.55$.

Note that, given a fixed erasure rate ϵ, for RLRS code the order of encoding complexity per information packet is $O(k)$. Fig. 1 shows typical encoding complexity curves versus the erasure rate ϵ with $q = 2^m = 256$ as the initial field size for two different values of k. The vertical axis in this figure shows the time required for encoding normalized by the number of information bits at the encoder. The encoding complexity of a Cauchy-based RS code that uses a field of size Q from the beginning is also shown. Note that these two codes exhibit the same failure probability for all values of ϵ. Further comparisons are provided later.

B. Decoding Complexity

After successfully receiving k encoded packets, the decoder first substitute the systematic packets in the rest of the equations. Then, the coefficient matrix of the remaining equations has a Cauchy form. We assume that the decoder simply performs matrix inversion to find the rest of the input packets. Inversion of a Cauchy matrix needs $O(k^2)$ field operations [20]. We soon show that the complexity of symbol-by-packet multiplications and additions is also $O(k^2)$. Thus, our RLRS codes have an overall decoding complexity of $O(k)$ per information packet. In comparison, random fountain codes have an overall $O(k^3)$ complexity (due to matrix inversion)[2], which results in $O(k^2)$ complexity per packet.

Assume that after e field extension the decoder has successfully received k encoded packets. If j packet out of the k received packets are systematic, for substituting these systematic packets in the system of equations, $j(k-j)$

[2]In practice, if the number of symbols in each packet is considerably larger than k, symbol-by-packet multiplications are the dominant portion of computations. In this case, the complexity of matrix inversion is negligible and both schemes have the same complexity $O(k^2)$.

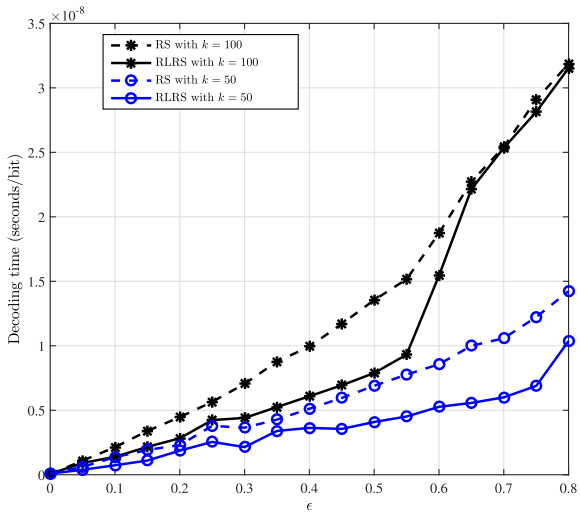

Fig. 2. Decoding time complexity per information bit for RLRS codes with initial field size $q = 256$ and Cauchy-based RS codes over a field of size $Q = q^2$. For $k = 50$, the effect of field extension becomes more pronounced when $\epsilon > 0.75$. The effect of field extension for $k = 100$, on the other hand, comes into play when $\epsilon > 0.55$.

packet additions and $j(k-j)$ symbol-by-packet multiplications over a finite field of size 2^{m2^e} should be performed. The decoder then obtains the inverse of coefficient matrix of the remaining system of equations (which has a Cauchy form) with $O\left((k-j)^2\right)$ computational complexity. Finally, the decoder multiplies the inverse matrix in the known packets to find the remaining source packets. This requires $j(k-j)$ packet additions and $j(k-j)$ symbol-by-packet multiplications. Thus, having j systematic packets in hand, the decoding complexity is

$$X = 2j(k-j) \times \left(2^{B-e}\mathcal{M}(2^e m) + 2^{B-e}\mathcal{A}(2^e m)\right) \quad (27)$$

$$= 2j(k-j) \times \frac{\chi(2^e m)}{k} = 2j(k-j)\frac{\chi(m)}{k}\xi^e. \quad (28)$$

The average number of systematic packets that the decoder receives is $j = (1-\epsilon)k$. Thus, the average decoding complexity per information packet for RLRS codes, C_D, is given by

$$C_D = 2(1-\epsilon)\epsilon\chi(m)\left(1 + \sum_{e=1}^{B}\left(P_f(e-1) - P_f(e)\right)\xi^e\right). \quad (29)$$

which for $B = 1$ simplifies to

$$C_D = 2(1-\epsilon)\epsilon\chi(m)\left(1 + \left(P_f(0) - P_f(1)\right)\xi\right)$$
$$< 2(1-\epsilon)\epsilon\chi(m)\left(1 + P_f(0)\xi\right). \quad (30)$$

Fig. 2 shows the complexity of decoding versus ϵ for some practical values of k obtained by real simulations of data transmissions. Again, the vertical axis illustrates the running time of the decoder normalized by the total number of transmitted information bits (i.e., k times length of each packet). This figure also illustrates the decoding complexity of a Cauchy-based RS code that uses a field of size Q from the beginning. As seen, when field extension is not performed, RLRS codes are significantly faster, but when field extension is performed there is no difference between the two solutions. In a setup, where there are users with a variety of erasure rate, RLRS can benefit users with better erasure rates.

VI. Discussion

In this part we first provide a high level comparison between our RLRS codes and other rateless coding solutions. Numerical comparisons between encoding and decoding complexity of RLRS codes and RaptorQ are also provided in Appendix II.

Erasure coding based on RS codes has been studied in the past [8], [9]. The problem with using an RS code as an erasure code is that it is not truly rateless. Suggested modifications to resolve this problem have sacrificed the MDS property, resulting in reception overhead. Our RLRS codes, however, are both MDS and rateless.

One may notice that there might be regions of operations that RS codes over $\mathbb{F}_{q'}$ with $q \le q' \le q^2$ are better candidates than having an RLRS code with initial field $\mathbb{F}_q$. However, if the user of the RS code has such an accurate channel knowledge to wisely chose q', it is only fair to assume that the user of the RLRS code also has the same knowledge. In this case the user of the RLRS code can also use q' for the initial field size and hence will never be worse than the user of the RS code. Using this channel knowledge, the RLRS code user may even choose a wiser initial field size than q'.

Another class of solutions for erasure coding is LT or Raptor coding, built upon the idea of low-complexity decoding. These codes are attractive solutions when k is large enough. Otherwise, for conventional LT codes have considerable reception overhead. For example, overheads in the range of 25 to 35 percent is expected when $k = 100$. To overcome this problem, use of non-binary fountain codes has been proposed [10], [12]. The most advance Raptor code, namely RaptorQ, uses a finite field of size 256 to mimic the performance of non-binary Random Linear Fountain codes [24]. RaptorQ, therefore, enjoys the benefits of Random Linear Fountain codes while maintains a low encoding and decoding complexity [12]. In an example provided in Appendix II we compare RaptorQ and RLRS codes in different settings.

We recognize that our RLRS codes have a higher coding complexity than LT or Raptor codes for large k. This is consistent with our analysis which shows that per-packet coding complexity of RLRS codes grows as $O(k)$ as opposed to $O(1)$ for Raptor codes. Therefore, the main application of our codes are for the cases where k is not too large. One should also notice that the complexity of our RLRS codes depends on the channel erasure rate (see Fig. 1 and 2.) Using RLRS codes in a multicast scenario, a user with a better channel quality can enjoy a lower complexity of decoding. On the other hand, a user with a higher erasure rate should perform a more complex decoding process. Nevertheless, all users are able to recover the data with zero overhead. For small erasure rates, a large number of systematic packets are received and the size of the remaining system of equations at the decoder is small. According to (12), (25) and

TABLE I

+	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

TABLE II

×	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

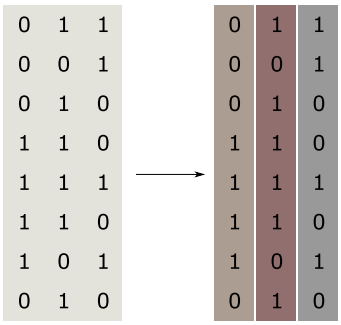

Fig. 3. A file of size $k \times m \times 2^B$ bits is divided to k source packets.

(30), for small ϵ's, both encoding and decoding complexities grow as $O(\epsilon k)$ per packet.

VII. Conclusion

In this paper, we presented a new class of rateless codes based on Cauchy-based RS codes. These codes are proposed for applications that have a small source block size. Using a new technique, called preservative field extension, a Cauchy-based RS code is turned to a rateless code. The preservative field extension lets the resulting codes be MDS and guarantees zero reception overhead over packet networks with erasure links. Additionally, the coding complexity of the resulting codes is investigated. We showed that the proposed coding scheme can achieve a lower coding complexity than other rateless coding solutions.

Appendix I
Encoding Example

Consider a source of information that has a file of size $k \times m \times 2^B$ and uses an RLRS code with design length $k = 3$ and initial field size $q = 2^m = 4$. The addition and multiplication tables for the initial field is given in Table I and II. Addition is simply the bitwise **XOR** of the corresponding bits. We assume that $B = 2$ meaning that each network packet contains 8 bits.

The encoder first breaks the file into 3 input packets as shown in Fig. 3.

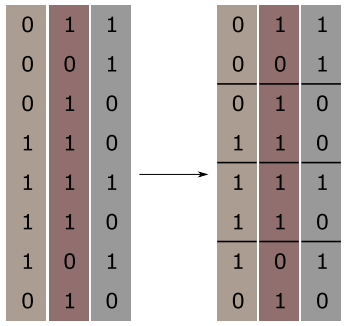

Fig. 4. Each source packet with length $m \times 2^B$ is considered as 2^B symbols in $\mathbb{F}_q$ with $q = 2^m$.

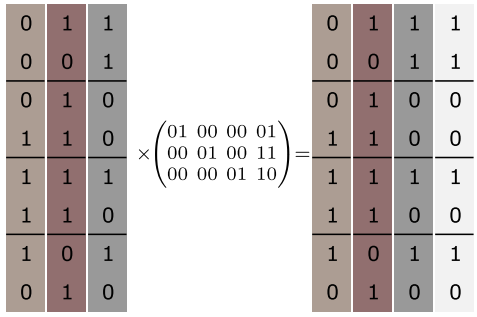

Fig. 5. Encoding of k source packets into q output packets over $\mathbb{F}_q$.

According to (1) and (2), the first four columns of the generator matrix in the initial field can be seen as

$$G_q(1:3,1:4) = \begin{pmatrix} 01 & 00 & 00 & \frac{01}{00+01} \\ 00 & 01 & 00 & \frac{01}{00+10} \\ 00 & 00 & 01 & \frac{01}{00+11} \end{pmatrix} = \begin{pmatrix} 01 & 00 & 00 & 01 \\ 00 & 01 & 00 & 11 \\ 00 & 00 & 01 & 10 \end{pmatrix} \quad (31)$$

where, by $G(a:b, c:d)$ we mean the $(b-a+1) \times (d-c+1)$ sub-matrix of G such that the element in its i^{th} row and j^{th} column is $g_{(a+i-1,c+j-1)}$.

When no field extension is performed, each information packet is treated as 4 symbols in $\mathbb{F}_q$ as shown in Fig. 4.

The first four encoded packets are formed by multiplying $G_q(1:3,1:4)$ by information packets as shown in Fig. 5.

Now if any set of 3 columns from $G_q(1:3,1:4)$ is successfully received, the decoder will be able to recover the original data. In case that two or more of the encoded packets are erased by the channel, the decoder does not have enough packets to decode the information. Hence, the encoder performs one field extension and continues to transmit encoded packets based on $G_Q(1:3,5:16)$ where $\mathbb{F}_Q$ is the preservative extension of $\mathbb{F}_q$ with $Q = q^2 = 16$. To construct $\mathbb{F}_Q$ from $\mathbb{F}_q$, we should first find an element $h \in \mathbb{F}_q$ that $\mathrm{Tr}(h) \neq 0$. It is straight forward to show that $h = 10$ is such an element. Using (7), the multiplication table for $\mathbb{F}_Q$ is given in Table III. Again, addition is simply the bitwise **XOR** of the corresponding bits.

Before showing how the encoder can generate new encoded packets, let us first show that the previously encoded packets

TABLE III
MULTIPLICATION TABLE FOR $\mathbb{F}_Q$

×	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0001	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0010	0000	0010	0011	0001	1000	1010	1011	1001	1100	1110	1111	1101	0100	0110	0111	0101
0011	0000	0011	0001	0010	1100	1111	1101	1110	0100	0111	0101	0110	1000	1011	1001	1010
0100	0000	0100	1000	1100	1111	1011	0111	0011	0101	0001	1101	1001	1010	1110	0010	0110
0101	0000	0101	1010	1111	1011	1110	0001	0100	1101	1000	0111	0010	0110	0011	1100	1001
0110	0000	0110	1011	1101	0111	0001	1100	1010	1001	1111	0010	0100	1110	1000	0101	0011
0111	0000	0111	1001	1110	0011	0100	1010	1101	0001	0110	1000	1111	0010	0101	1011	1100
1000	0000	1000	1100	0100	0101	1101	1001	0001	1010	0010	0110	1110	1111	0111	0011	1011
1001	0000	1001	1110	0111	0001	1000	1111	0110	0010	1011	1100	0101	0011	1010	1101	0100
1010	0000	1010	1111	0101	1101	0111	0010	1000	0110	1100	1001	0011	1011	0001	0100	1110
1011	0000	1011	1101	0110	1001	0010	0100	1111	1110	0101	0011	1000	0111	1100	1010	0001
1100	0000	1100	0100	1000	1010	0110	1110	0010	1111	0011	1011	0111	0101	1001	0001	1101
1101	0000	1101	0110	1011	1110	0011	1000	0101	0111	1010	0001	1100	1001	0100	1111	0010
1110	0000	1110	0111	1001	0010	1100	0101	1011	0011	1101	0100	1010	0001	1111	0110	1000
1111	0000	1111	0101	1010	0110	1001	0011	1100	1011	0100	1110	0001	1101	0010	1000	0111

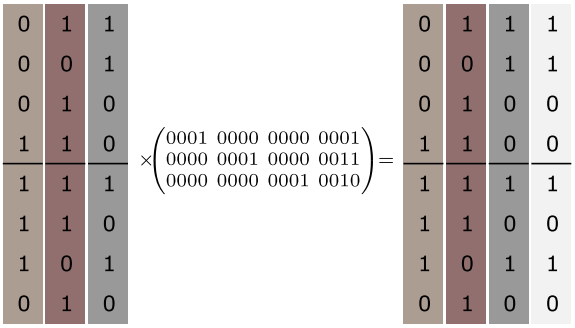

Fig. 6. Encoding of k source packets into q output packets over $\mathbb{F}_Q$.

in $\mathbb{F}_q$ are still valid in $\mathbb{F}_Q$. By having a look on the first 4 columns of G_Q, we see that the encoder could use this sub-matrix in the extended field instead of $G_q(1:3,1:4)$. The equivalent elements of $G_Q(1:3,1:4)$ can be obtained by simply zero-padding of the elements of $G_q(1:3,1:4)$ on the left:

$$G_q(1:3,1:4) = \begin{pmatrix} 0001 & 0000 & 0000 & \frac{0001}{0000+0001} \\ 0000 & 0001 & 0000 & \frac{0001}{0000+0010} \\ 0000 & 0000 & 0001 & \frac{0001}{0000+0011} \end{pmatrix} = \begin{pmatrix} 0001 & 0000 & 0000 & 0001 \\ 0000 & 0001 & 0000 & 0011 \\ 0000 & 0000 & 0001 & 0010 \end{pmatrix} \quad (32)$$

As shown in Fig. 6, had the encoder used this matrix for the first 4 encoded packets, it would have generated the same encoded packets as before. This is easily verified by comparing the fourth encoded packets in Fig. 5 and Fig. 6, where multiplications in Fig. 4 are based on Table III.

Now, let us discuss encoding of new packets. The sub-matrix $G_Q(1:3,5:16)$ is given by

$$G_Q(1:3,5:16) = \begin{pmatrix} \frac{0001}{0100+0001} & \frac{0001}{0101+0001} & \frac{0001}{0110+0001} & \cdots & \frac{0001}{1111+0001} \\ \frac{0001}{0100+0010} & \frac{0001}{0101+0010} & \frac{0001}{0110+0010} & \cdots & \frac{0001}{1111+0010} \\ \frac{0001}{0100+0011} & \frac{0001}{0101+0011} & \frac{0001}{0110+0011} & \cdots & \frac{0001}{1111+0011} \end{pmatrix}. \quad (33)$$

Note that the Cauchy structure of the generator matrix has been successfully preserved and the code is still MDS.

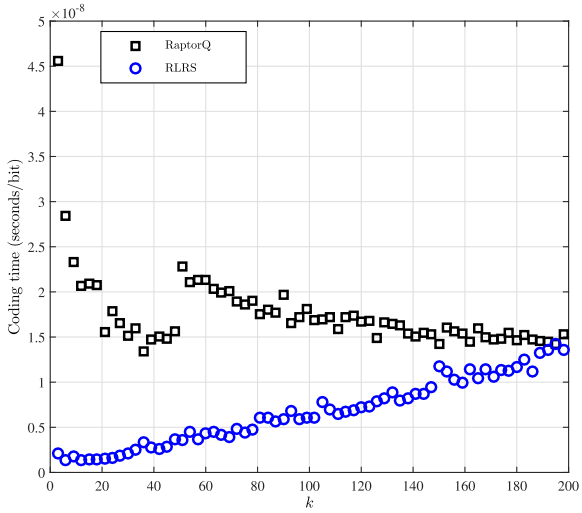

Fig. 7. Time complexity of encoding and decoding per information bit for RaptorQ and RLRS codes with initial field size $q = 256$ when $\epsilon = 0.2$.

The encoder now continues transmission in $\mathbb{F}_Q$ and the decoder will listen and waits until it collects 4 encoded packets. In this case, the decoder treats all the received packets as output packets of an encoder with generator matrix G_Q. If needed, another field extension can be performed.

As seen in Fig. 6, the packet size is not affected after the field extension is performed. However, the number of symbols in each packet is decreased.

APPENDIX II
NUMERICAL COMPARISONS OF RLRS AND RAPTORQ CODES

In this example, we compare the required time for encoding and decoding of RLRS codes and RaptorQ codes. The initial field size of RLRS code is set to $q = 2^8$. Fig. 7 to 10 illustrate the coding time for both RLRS and RaptorQ codes All simulations regarding RLRS codes have been implemented in C++. Simulations are done on a Macbook Pro with 2.8 GHz Dual-core Intel Core i7(Turbo Boost up to 3.3GHz) and 16 GB 1600 MHz DDR3L SDRAM. For RaptorQ codes, we obtained

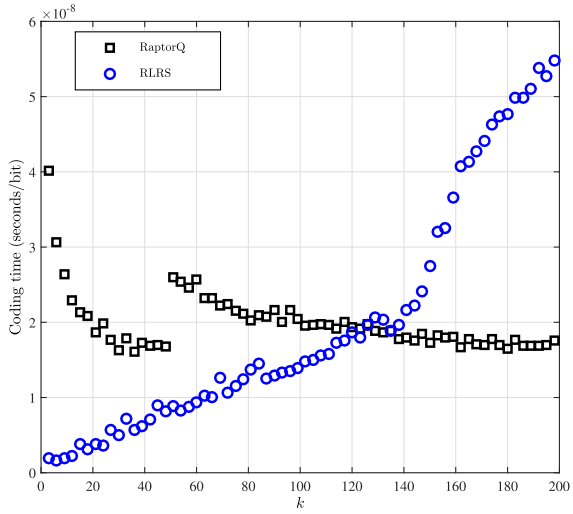

Fig. 8. Time complexity of encoding and decoding per information bit for RaptorQ and RLRS codes with initial field size $q = 256$ when $\epsilon = 0.4$.

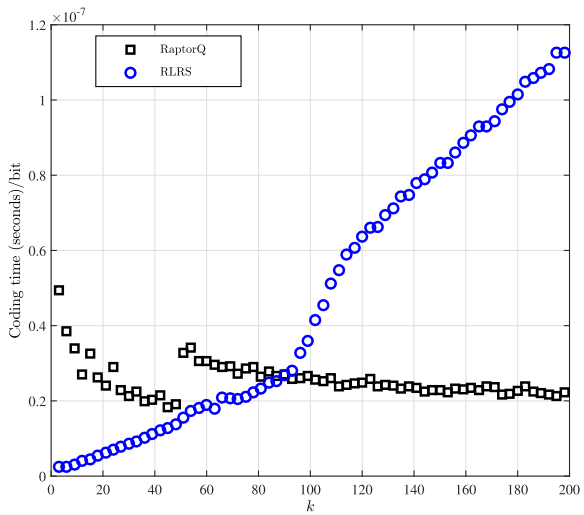

Fig. 9. Time complexity of encoding and decoding per information bit for RaptorQ and RLRS codes with initial field size $q = 256$ when $\epsilon = 0.6$.

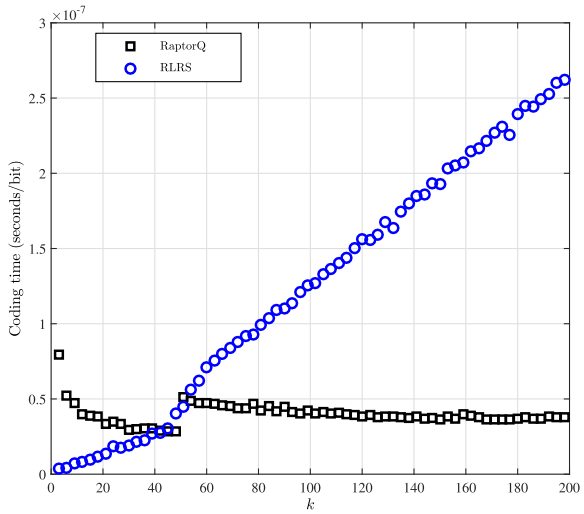

Fig. 10. Time complexity of encoding and decoding per information bit for RaptorQ and RLRS codes with initial field size $q = 256$ when $\epsilon = 0.8$.

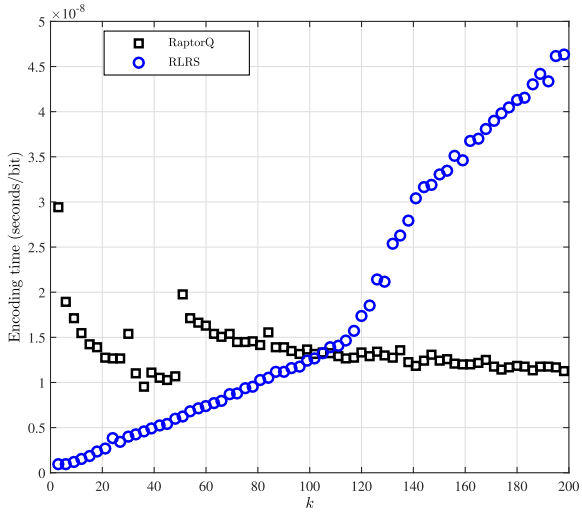

Fig. 11. Time complexity of encoding per information bit for RaptorQ and RLRS codes with initial field size $q = 256$ when $\epsilon = 0.5$.

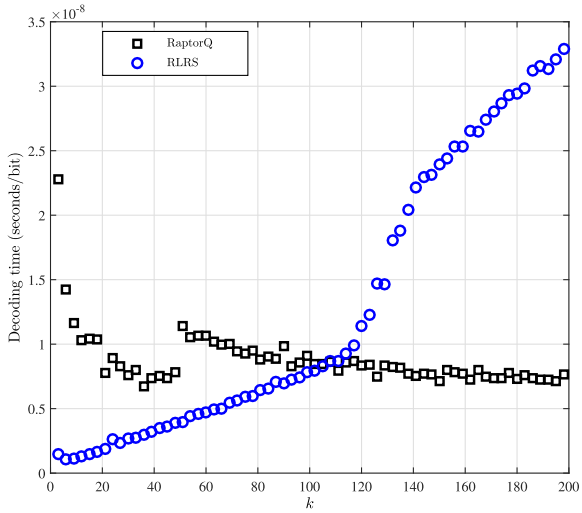

Fig. 12. Time complexity of decoding per information bit for RaptorQ and RLRS codes with initial field size $q = 256$ when $\epsilon = 0.5$.

the Qualcomm® RaptorQ™ Evaluation Kit and compared coding time of our RLRS codes with the results of this kit. Here, by coding time we mean the running time of the encoder plus the running time of decoder normalized by the total number of information bits. These figures show that, as expected, the complexity of our RLRS codes is higher than that of RaptorQ codes for large values of k. This is due to $O(k)$ versus $O(1)$ per-packet complexity growth of these codes. However, according to Fig. 7 and Fig. 8, when k is not too large or when the channel does not suffer from a large erasure rate (i.e., ϵ is not too large) RLRS codes perform better than RaptorQ codes. For both RLRS and RaptorQ codes, encoding time and decoding time exhibit the same trend. In order to better separate the portion of complexity for encoder and decoder, Fig. 11 and Fig. 12 illustrate the encoding and decoding time of RLRS and RaptorQ codes separately when $\epsilon = 0.5$. As explained earlier, this two figures verify that RLRS codes might be better candidates for forward error correction over the packet erasure channel when k is not too large.

ACKNOWLEDGMENT

The authors acknowledge Qualcomm[3] for providing their RaptorQ evaluation kit that is used in this study.

REFERENCES

[1] J. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1528–1540, Oct. 2002.
[2] M. Luby, "LT codes," in *Proc. 43rd Annu. IEEE Symp. Found. Comput. Sci.*, 2002, pp. 271–280.
[3] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.
[4] D. MacKay, "Fountain codes," *IEE Proc. Commun.*, vol. 152, no. 6, pp. 1062–1068, Dec. 2005.
[5] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, 1960.
[6] S. B. Wicker, *Reed-Solomon Codes and Their Applications*. Piscataway, NJ, USA: IEEE Press, 1994.
[7] A. Soro and J. Lacan, "FNT-based Reed-Solomon erasure codes," in *Proc. 7th IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2010, pp. 1–5.
[8] L. Rizzo and L. Vicisano, "A reliable multicast data distribution protocol based on software FEC techniques," in *Proc. 4th IEEE Workshop High-Perform. Commun. Syst. (HPCS'97)*, 1997, pp. 116–125.
[9] L. Vicisano, J. Crowcroft, and L. Rizzo, "TCP-like congestion control for layered multicast data transfer," in *Proc. IEEE INFOCOM*, Mar. 1998, pp. 996–1003, vol. 3.
[10] G. Liva, E. Paolini, and M. Chiani, "Performance versus overhead for fountain codes over $\mathbb{F}_q$," *IEEE Commun. Lett.*, vol. 14, no. 2, pp. 178–180, Feb. 2010.
[11] B. Schotsch, R. Lupoaie, and P. Vary, "The performance of low-density random linear fountain codes over higher order Galois fields under maximum likelihood decoding," in *Proc. 49th Annu. Allerton Conf. Commun. Control Comput.*, Sep. 2011, pp. 1004–1011.
[12] A. Shokrollahi and M. Luby, "Raptor codes," *Found. Trends Commun. Inf. Theory*, vol. 6, no. 3, pp. 213–322, 2009.
[13] S. B. Wicker, *Reed-Solomon Codes and Their Applications*. Piscataway, NJ, USA: IEEE Press, 1994.
[14] E. R. Berlekamp, *Algebraic Coding Theory*. New York, NY, USA: McGraw-Hill, 1968.
[15] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inf. Theory*, vol. 15, no. 1, pp. 122–127, Jan. 1969.
[16] N. B. Atti, G. M. Diaz-Toca, and H. Lombardi, "The Berlekamp-Massey algorithm revisited," *Appl. Algebra Eng. Commun. Comput.*, vol. 17, no. 1, pp. 75–82, Apr. 2006.
[17] R. Chien, "Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes," *IEEE Trans. Inf. Theory*, vol. 10, no. 4, pp. 357–363, Oct. 1964.
[18] G. Forney, "On decoding BCH codes," *IEEE Trans. Inf. Theory*, vol. 11, no. 4, pp. 549–557, Oct. 1965.
[19] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometry codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 1757–1767, Sep. 1999.
[20] J. Blomer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman. (1995). *An XOR-Based Erasure-Resilient Coding Scheme* [Online]. Available: http://www1.icsi.berkeley.edu/~luby
[21] J. S. Plank and L. Xu, "Optimizing Cauchy Reed-Solomon codes for fault-tolerant network storage applications," in *Proc. 5th IEEE Int. Symp. Netw. Comput. Appl. (NCA0'6)*, 2006, pp. 173–180.
[22] R. J. McEliece, *Finite Field for Scientists and Engineers*. Norwell, MA, USA: Kluwer, 1987.
[23] J. Cherly, L. Gallardo, L. N. Vaserstein, and E. Wheland, "Solving quadratic equations over polynomial rings of characteristic two," *Publ. Mat.*, vol. 42, no. 1, pp. 131–142, 1998.
[24] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder. (2011, Aug.). *RaptorQ Forward Error Correction Scheme for Object Delivery*, RFC 6330 (Proposed Standard), Internet Engineering Task Force [Online]. Available: http://www.ietf.org/rfc/rfc6330.txt

Reza Rafie Borujney (S'14) received the B.Sc. degree from the University of Tehran, Tehran, Iran, and the M.Sc. degree from the University of Alberta, Edmonton, AB, Canada, in 2012 and 2014, respectively. He is currently pursuing the Ph.D. degree in electrical engineering at the University of Toronto, Toronto, ON, Canada. His research interests include coding theory.

Masoud Ardakani (M'04–SM'09) received the B.Sc. degree from Isfahan University of Technology, Isfahan, Iran, the M.Sc. degree from the University of Tehran, Tehran, Iran, and the Ph.D. degree from the University of Toronto, Toronto, ON, Canada, all in electrical engineering, in 1994, 1997, and 2004, respectively. He was a Postdoctoral Fellow with the University of Toronto from 2004 to 2005. He is currently a Professor of electrical and computer engineering with the University of Alberta, Edmonton, AB, Canada. His research interests include digital communications. He serves as an Associate Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS and has served as an Associate Editor for the IEEE WIRELESS COMMUNICATIONS and a Senior Editor for the IEEE COMMUNICATION LETTERS.

[3]Qualcomm is a trademark of Qualcomm Incorporated, registered in the United States and other countries, used with permission.