An Efficient (n, k) Information Dispersal Algorithm for High Code Rate System over Fermat Fields

Sian-Jheng Lin and Wei-Ho Chung

***Abstract*—The (n, k) information dispersal algorithm (IDA) is the art of converting a file into n pieces of shadows, and any k out of the n shadows suffice for reconstructing the file. The IDA is applicable to the distributed communication and storage systems. This letter proposes an efficient (n, k) IDA for the case $n/2 \leq k < n$ over Fermat field GF(2^r+1). We first present the IDA under conditions of $n-k$ in the power of 2 and n as multiple of $n-k$, and then extend the algorithm to the general case. For a reasonably large file, both encoder and decoder achieve $\Theta(n\log(n-k))$ operations in processing k symbols.**

***Index Terms*—Erasure codes, fast Fourier transforms, information dispersal algorithm (IDA), Reed-Solomon codes.**

I. Introduction

THE coding for erasure channel is crucial for distributed storages [11] and data transmission systems [10]. Information dispersal algorithm (IDA) is a type of erasure coding proposed by Rabin [1], [2] in 1989. The (n, k) IDA is used to disperse a file into n shadows so that any k out of the n shadows can jointly reconstruct the information of the file. The complexity is one of the most critical issues for the IDA, particularly in the real-time communication or storage systems demanding large amount of encoding and decoding operations. Several applications of erasure codes on networks are given in [10], such as multicast, parallel downloading, and one-to-many TCP.

Based on Fermat number transforms over Fermat field GF(2^r+1), we concentrate on proposing a (n, k) IDA for the case $n/2 \leq k < n$ in this letter. For a reasonably large file, both encoder and decoder achieve $\Theta(n\log(n-k))$ operations in processing k symbols. In our survey of the literature, this is a new complexity bound for $k \geq n/2$. Since the proposed algorithm is an implementation of erasure code, the algorithm can be applied on several applications based on the erasure code techniques [1], [2], [10].

The rest of this letter is organized as follows. Section II describes the coding system and the environment settings. Section III and Section IV respectively introduce the coding algorithms for $n-k$ in the power of 2 and n as multiple of $n-k$. Section V extends the (n, k) to the general case. Section VI shows the complexity comparisons with other existing works. Section VII discusses the criteria of applying the proposed IDA or the ordinary method.

Manuscript received June 15, 2012. The associate editor coordinating the review of this letter and approving it for publication was A. Burr.

The authors are with the Research Center for Information Technology Innovation, Academia Sinica, Taipei City, Taiwan (e-mail: {sjlin, whc}@citi.sinica.edu.tw)

This work was supported by the National Science Council of Taiwan, under grant number NSC101-2221-E-001-002, NSC101-2221-E-001-008, and 32T-1010721-1C.

Digital Object Identifier 10.1109/LCOMM.2012.112012.121322

II. Coding system

The code system is a modified version of the systematic Reed-Solomon codes over Fermat field. The Fermat field is defined as the finite field GF(2^r+1) for $r \in \{1, 2, 4, 8, 16\}$. The computations and code symbols are over Fermat field throughout this letter. We denote $M_m(x_1, x_2 \ldots x_n)$ as a $m \times n$ Vandermonde matrix whose i-th row is $[x_1^{i-1}\, x_2^{i-1} \ldots x_n^{i-1}]$. Given k input symbols $\{a_i | i=1 \text{ to } k\}$ taken from the source file F, the first k symbols of the n-element codeword $v=[v_1\ v_2 \ldots v_n]^T$ are $\{v_i=a_i | i=1 \text{ to } k\}$, and the remaining $m=n-k$ parity symbols are computed through solving the m equations:

$$M_m(I_1, I_2 \ldots I_n) \times v = \mathbf{0}, \tag{1}$$

where $\mathbf{0}$ is a zero vector, and each element I_i, called the evalutaion point of v_i, has the property $I_i \neq I_j\ \forall i \neq j$. In decoding, the decoder receives a codeword $r=[r_1\ r_2 \ldots r_n]^T$ with m erasures. Let the set $L=\{l_i | i=1 \text{ to } m\}$ denote the locations of m erasures; then for each $j \in L$, the received symbol $r_j=0$, or else $r_j=v_j$. At first, the syndrome $s=[s_1\ s_2 \ldots s_m]^T$ is computed by

$$s = M_m(I_1, I_2 \ldots I_n) \times [-r_1 - r_2 \ldots - r_n]^T. \tag{2}$$

Then the erasure values $e=[e_{l1}\ e_{l2} \ldots e_{ln}]^T$ are determined by solving m linear equations:

$$M_m(I_{l_1}, I_{l_2} \ldots I_{l_m}) \times e = s. \tag{3}$$

Since the matrix $M_m(I_{l1}, I_{l2} \ldots I_{lm})$ is invertible, the erasure values can be uniquely solved.

Based on the transformation described above, the proposed IDA transforms the file F through (1) into n shadows $\{(I_i, F_i) | i=1 \text{ to } n\}$, where each F_i contains the codeword symbol v_i corresponding to the evaluation point I_i. In 2^r+1-ary numeral system, the length of each shadow F_i is one-k^{th} length of the file F. However, the 2^r+1-ary symbol is unable to be directly coded in a r-bit unit. Soro and Lacan [6] suggest that, for symbols in the range of the 1st and 2^r-th values, the symbol is stored in r-bit unit. For the overflowing (2^r+1)-th value, the header of shadow file records the positions indicating the occurrence of the overflowing (2^r+1)-th value in the 2^r+1-ary numeral system. In Sec. III and Sec. IV, we begin by constructing the IDA under conditions m being power of 2, and k being multiple of m. In Sec. V, we extend the presented IDA to the general case.

The proposed IDA employs the fast Fermat number transforms (FNT), which is equivalent to the fast Fourier transforms over Fermat field. Given a vector x with length m being power of two, the m-points FNT and the inverse (IFNT) are expressed as

$$X = FNT_m(x), \text{ and } x = IFNT_m(X), \tag{4}$$

which require $\Theta(m\log m)$ operations with m-points FFT over finite field. The twiddle factors $\{W_m^i|\forall i\}$ used in the FFT butterfly can be calculated through $W_m=\alpha^{2^r/m}$, where α is the primitive element of Fermat field. For simplicity, the notations FNT_m and IFNT_m respectively denote the m-points FNT and IFNT, and the notation $\phi(m)$ denotes the operation counts of the FNT_m or IFNT_m.

For the value of I_l, the position l divided by m to obtain the quotient j and the remainder i, then the value of evaluation point is defined as

$$I_l = I_{i+mj} = \alpha^j W_m^i, 0 \le i \le m-1, 0 \le j \le n/m-1. \quad (5)$$

III. Encoder

The encoder requires n/m times of FNT_m. The k input symbols are divided into k/m sectors $v_j=[a_{mj}\ a_{1+mj}\dots a_{m-1+mj}]^T$ with length m, and the last sector $v_{k/m}=[b_1\ b_2\dots b_m]^T$ consisting of the parity symbols is our object to be calculated. Then (1) is reformulat

$$\mathbf{0} = \sum_{j=0}^{k/m} M_m(\alpha^j W_m^0, \alpha^j W_m^1 ..., \alpha W_m^{m-1}) \times v_j. \quad (6)$$

The $m\times m$ sub-matrix can be expressed as

$$\begin{aligned} &M_m(\alpha^j W_m^0, \alpha^j W_m^1 \dots \alpha^j W_m^{m-1}) \\ &= diag(1, \alpha^j \dots \alpha) \times M_m(W_m^0, W_m^1 \dots W_m^{m-1}), \end{aligned} \quad (7)$$

where $diag(X)$ denotes a diagonal matrix whose diagonal entries are the elements of the vector X. Since $M_m(W_m^0, W_m^1, ..., W_m^{m-1})$ is an $m\times m$ Fourier matrix, the FNT_m can be employed. Let

$$b_j = (1, \alpha^j, \dots, \alpha^{j(m-1)}) \quad (8)$$

denote a m-point vector, and the symbol $\otimes$ denote the dot product operation. Then (6) is rewritten as

$$\begin{aligned} &\mathbf{0} = \sum_{j=0}^{k/m} b_j \otimes FNT_m(v_j) \\ &\Rightarrow -b_{k/m} \otimes FNT_m(v_{k/m}) = \sum_{j=0}^{k/m-1} b_j \otimes FNT_m(v_j) \\ &\Rightarrow v_{k/m} = IFNT_m(\sum_{j=0}^{k/m-1} b_j^* \otimes FNT_m(v_j)), \end{aligned} \quad (9)$$

where $b_j^* = -b_j/b_{k/m} = (-1, -\alpha^{j-k/m} \dots -\alpha^{(j-k/m)(m-1)})$. The encoding algorithm is provided below:

Input: k data symbols $\{a_i|i=1 \text{ to } k\}$.
Output: m parity symbols $\{b_i|i=1 \text{ to } m\}$.
Main procedure:

1) Compute the vectors $V_j=FNT_m(a_{mj}, a_{1+mj} \dots a_{m-1+mj})$ for $j = 0$ to $k/m-1$.
2) Compute the vector $V_{k/m} = \sum_{j=0}^{k/m-1} b_j^* \otimes V_j$.
3) Compute $v_{n/m} = IFNT_m(V_{k/m})$.

In the first step, computing $\{V_j|j=0 \text{ to } k/m-1\}$ requires $n/m-1$ times of FNT_m. In the second step, computing $V_{k/m}$ requires $n/m-1$ times of m-points dot products and $n/m-2$ times of m-points vector summations. In the last step, an IFNT_m is required. Thus, the procedure requires overall $n/m\times\phi(m) + 2n-3m = \Theta(n\log m)$ operations.

IV. Decoder

The proposed decoder follows the Forney algorithm, which is a method to compute the error values of BCH codes at known error locations. We firstly introduce the concept of Forney algorithm [3], and then explain the proposed algorithm.

A. Forney algorithm

When decoder receives a codeword with m erasures, in the first step, the syndrome is computed through (2), and we define the syndrome polynomial as

$$S(x) = \sum_{i=1}^{m} s_i x^{i-1}. \quad (10)$$

In the second step, the error location polynomial is defined as

$$\Lambda(x) = \prod_{i=1}^{m}(1 - I_{l_i} x) = 1 + \sum_{i=1}^{m} \lambda_i x^i. \quad (11)$$

Then the error evaluator polynomial is computed through

$$Z(x) = S(x)\Lambda(x) \,(\text{mod}\, x^m) = \sum_{i=0}^{m-1} z_i x^i. \quad (12)$$

In the last step, the erasure values are calculated through

$$e_{l_i} = (-I_{l_i} \times \Lambda'(I_{l_i}^{-1})^{-1}) \times Z(I_{l_i}^{-1}), \text{for } i = 1 \text{ to } m. \quad (13)$$

The formal derivative of $\Lambda(x)$ is defined as

$$\Lambda'(x) = \sum_{i=1}^{m} i \times \lambda_i x^{i-1}. \quad (14)$$

B. Decoding algorithm

The decoding algorithm is based on two observations. Firstly, since the erasure locations of all received codewords are identical, the calculations requiring only the erasure locations can be completed in one round without repetitive calculations in later iterations. This part is called the initialization of the decoder. Secondly, we employ the FNT_m in main procedure to reduce the calculation overhead as much as possible. In the following, we describe our algorithm step-by-step by following the Forney algorithm.

In the first step, the syndrome is calculated by the FNT_m. The received codeword is divided into k/m sectors $r^j=(r_{mj}, r_{1+mj} \dots r_{m-1+mj})$ with length m. Then the syndrome $s=(s_1, s_2 \dots s_m)$ is computed through

$$s = -\sum_{j=0}^{k/m} b_j \otimes FNT_m(r^j). \quad (15)$$

In the second step, since the error location polynomial (11) requires only information of the erasure locations, the $\lambda=(1, \lambda_1, \lambda_2 \dots \lambda_m)$ is calculated in the initialization. Then the (12) is calculated by fast polynomial multiplication. Let two $2m$-element vectors be $s^*=(s_1, s_2 \dots s_m, 0 \dots 0)$ and $\lambda^*=(1, \lambda_1, \lambda_2 \dots \lambda_m, 0 \dots 0)$. Then the coefficients $z=(z_1, z_2 \dots z_m)$ are the left half elements of

$$z^* = IFNT_{2m}(FNT_{2m}(s^*) \otimes FNT_{2m}(\lambda^*)). \quad (16)$$

Since the computation of $FNT_{2m}(\lambda^*)$ requires only the locations of erasures, the $\Lambda = FNT_{2m}(\lambda^*)$ can be calculated in the initialization. Furthermore, we observe that several redundant operations in (16) can be removed by decomposing the FNT_{2m} into two FNT_m. To do so, two m-points vectors Λ_0 and Λ_1 respectively denote the even and odd indices of Λ, and two m-point vectors $c = (W_{2m}^0, W_{2m}^1 \dots W_{2m}^{m-1})$ and $c^{-1} = (W_{2m}^0, W_{2m}^{-1} \dots W_{2m}^{-(m-1)})$ denote the twiddle factors used in the last step of FNT_{2m} butterfly. Then the $z=(z_1, z_2 \dots z_m)$ is calculated through

$$\begin{aligned} z = &IFNT_m(FNT_m(s) \otimes \Lambda_0) + \\ &c^{-1} \otimes IFNT_m(FNT_m(c \otimes s) \otimes \Lambda_1). \end{aligned} \quad (17)$$

In the last step, we define two polynomials by reversing the order of coefficients in $Z(x)$ and $\Lambda'(x)$:

$$\begin{aligned} Z_1(x) &= x^{m-1}Z(x^{-1}) \\ &= x^{m-1}\textstyle\sum_{i=0}^{m-1} z_i x^{-i} = \sum_{i=0}^{m-1} z_{m-i-1}x^i, \end{aligned} \tag{18}$$

$$\begin{aligned} \Lambda_1'(x) &= x^{m-1}\Lambda'(x^{-1}) \\ &= x^{m-1}\textstyle\sum_{i=1}^{m} i\lambda_i x^{-(i-1)} = \sum_{i=0}^{m-1}(m-i)\lambda_{m-i}x^i. \end{aligned} \tag{19}$$

Then the (13) is rewritten as

$$\begin{aligned} e_{l_i} &= (-I_{l_i} \times \Lambda'(I_{l_i}^{-1})^{-1}) \times Z(I_{l_i}^{-1}) \\ &= [-I_{l_i} \times (I_{l_i}^{1-m}\Lambda_1'(I_{l_i}))^{-1}] \times I_{l_i}^{1-m} \times Z_1(I_{l_i}) \\ &= (-I_{l_i} \times \Lambda_1'(I_{l_i})^{-1}) \times Z_1(I_{l_i}) \; \forall i = 1 \text{ to } m. \end{aligned} \tag{20}$$

According to the definition of indices (5), each erasure position can be expressed as l_i=q+mj, so the erasure evaluation point is

$$I_{l_i} = \alpha^j W_m^q, \forall l_i \in L \text{ and } l_i = q + mj. \tag{21}$$

We respectively plug (21) into (18)(19) to obtain

$$Z_1(I_{l_i}) = Z_1(\alpha^j W_m^q) = \textstyle\sum_{i=0}^{m-1}(z_{m-i-1}\alpha^{ji})W_m^{iq}, \tag{22}$$

$$\Lambda_1'(I_{l_i}) = \Lambda_1'(\alpha^j W_m^q) = \textstyle\sum_{i=0}^{m-1}[(m-i)\lambda_{m-i}\alpha^{ij}]W_m^{iq}. \tag{23}$$

Both equations are the Fourier transforms with shifting α^j. The vector $\omega_1 = (z_{m-1}, z_{m-2} \ldots z_0)$ expresses the coefficients of $Z_1(x)$, and the m-element vector λ_1' expresses the coefficients of $\Lambda_1'(x)$. For $j \in [0, n/m-1]$, we compute

$$g_j = FNT_m(b_j \otimes \omega_1), h_j = FNT_m(b_j \otimes \lambda_1'). \tag{24}$$

The value $Z_1(I_{l_i})$ is at the position k of the vector g_j, and the $\Lambda_1'(I_{l_i})$ is at the position k of the vector h_j. The j in (24) has at most n/m possible values, so the (24) requires at most n/m times of FNT_m to calculate g_j and h_j, respectively. It is noted that the calculation of $\Lambda_1'(I_{l_i})$ requires only the erasure locations, so those values

$$d = \{d_{l_i} = -I_{l_i} \times \Lambda_1'(I_{l_i})^{-1} | i = 1 \text{ to } m\} \tag{25}$$

can be computed in initialization. The decoding algorithm is summarized below:

Input: A codeword r with m erasures.

Output: m erasure values $\{e_{li} | i=1 \text{ to } m\}$.

Initialization: Compute the coefficients of $\Lambda(x)$, the $2m$-points vector Λ=FNT$_{2m}(\lambda')$, and the set d=$\{d_{li} | i = 1 \text{ to } m\}$ in (25).

Main procedure:

1) Compute the syndrome vector (15).
2) Compute the m-points vector z (17).
3) Compute (22) to obtain $Z_1(I_{li})$ for each $j \in \{l_i/m |\ l_i \in L$ and $i = 1$ to $m\}$. Then compute e_{li}=$d_{li} \times Z_1(I_{li})$, for i=1 to m.

In the initialization, computing the coefficients of $\Lambda(x)$ requires $\Theta(k\log^2 k)$ operations by the divide-and-conquer algorithm [4], or $\Theta(n\log n)$ by the fast convolution [5]. Computing the vector Λ needs a FNT$_{2m}$. For the construction of the set d, deriving $\Lambda(x)$ only takes m operations, and the (24) requires at most n/m times of FNT_m. Then we calculate each d_{li} by (25). Thus, the overall cost of the initialization is about $\Theta(k\log^2 k)$ + $\Theta(m)$ + $(n/m)\phi(m)$ + $\Theta(m)$ = $\Theta(k\log^2 k + n\log m)$.

In the main procedure, the first step requires n/m times of FNT$_m$, $n/m-1$ times of m-points dot products (because b_0 is an all-ones vector), and $n/m-1$ times of m-points vector summations. The second step requires two times of FNT$_m$, two times of IFNT$_m$, four times of m-points dot products, and a m-point vector addition. The last step requires at most n/m times of FNT$_m$, and m times of multiplications. Thus, the overall cost of the main procedure requires at most $(4+2n/m)\phi(m) + 2n+4m = \Theta(n\log m)$ operations.

V. IDA for General (n, k) Parameters

Sections III and IV present the IDA for m being power of 2, and n being multiple of m. Given the parameters (n', k') with $n'/2 \le k' < n'$ and $m' = n' - k'$, if the (n', k') does not precisely satisfy the above two conditions, the new (n, k) described in the following can be adopted in our coding algorithm:

$$m = 2^{\lceil \log_2 m' \rceil}, k = k'/m \times m, \text{ and } n = m + k. \tag{26}$$

In encoding, the encoder reads k input symbols concatenating k' message symbols with $k - k'$ zeros, resulting in $n-k$ parity symbols. Then the k'-symbol codeword is the combination of k' message symbols with $n' - k'$ parity symbols. In decoding, the decoder side receives k' symbols, and the $k - k'$ zeros are placed at the corresponding locations of the message part. Then the received codeword has k known values, so the $n-k$ erasures can be decoded.

VI. Comparisons

Table I lists the asymptotic complexities with the leading terms of prior FNT-based algorithms [6], [7], [8], [9] and ours. The ordinary method denotes the matrix multiplication strategy, where the encoder computes the parity symbols through multiplying k input symbols by a $k \times m$ matrix, and the decoder computes the erasures through multiplying the k received symbols by a $k \times m$ matrix. We briefly introduce those codes in the following.

We firstly introduce the non-systematic codes of [6], [8], [9]. For encoding, the [6], [8], [9] use a FNT$_n$ to transform k data symbols into n shadow symbols. For decoding, the [8] computes the $n-k$ un-received symbols by matrix multiplications, and [9] computes a recursion formula (see eq. (8) of [9]) within $\Theta((n-k)k)$ operations. Thus, the decoders [8], [9] contain a $\Theta(nm)$ or $\Theta(km)$ factors. The decoder of [6] is derived from Lagrange interpolation, and then [6] presents a fast polynomial multiplications to calculate the coefficients with 8 FNT$_n$s.

Regarding the systematic codes of [6], [7], [9], the [6] uses the non-systematic decoder of [6] to obtain the k intermediate symbols, and then apply a FNT$_n$ to obtain the resulting symbols. The [7] introduces a systematic code by using fast polynomial evaluation in encoding and fast polynomial interpolation in decoding. In Table I, the $\epsilon(n) = \Theta(n\log^2 n)$ denotes the overhead of evaluating n-point polynomial, and the $\iota(n) = \Theta(n\log^2 n)$ denotes the overhead of n-point polynomial interpolation. The [9] employs ordinary polynomial division to calculate the $n-k$ parity symbols without fast transforms; the decoder of [9] uses the non-systematic decoder [9] to obtain the k intermediate symbols, and then applies a FNT$_n$ to obtain the resulting symbols.

TABLE I: COMPARISONS WITH OTHER WORKS

	Syst.	Enc. Complex.	Dec. Complex.
Ord.	Y	$(2k-1)m=\Theta(km)$	$(2k-1)m=\Theta(km)$
[6]	N	$\phi(n)=\Theta(n\log n)$	$8\phi(n)=\Theta(n\log n)$
	Y	$9\phi(n)=\Theta(n\log n)$	$9\phi(n)=\Theta(n\log n)$
[7]	Y	$\phi(k)+\epsilon(max\{k,m\})$ $=\Theta(klog^2k)$	$\phi(k)+\iota(k)=\Theta(k\log^2 k)$
[8]	N	$\phi(n)=\Theta(n\log n)$	$2nm+\phi(n)=\Theta(n(m+\log n))$
[9]	N	$\phi(n)=\Theta(n\log n)$	$\phi(n)+2km=\Theta(n\log n+km)$
	Y	$2km=\Theta(km)$	$2\phi(n)+2km=\Theta(n\log n+km)$
Ours	Y	$n/m\times\phi(m)=\Theta(n\log m)$	$(4+2n/m)\phi(m)=\Theta(n\log m)$

To further compare our method with those algorithms, we implement the ordinary method, the [9] and ours for systematic case, and the [8] for non-systematic case. We also test the systematic case of [6] implemented by Soro and Lacan. To mitigate the influence of coding variations in the experiment, we employ the FFT code [6] on other programs [8], [9] and ours. Thus, all tested programs operate on the same version of FFT code. Those algorithms are tested on Intel Pentium B960 2.20 GHz×2 with 64-bits Ubuntu 12.04. Figure 1 depicts the results of the encoding and decoding for n=8192, and m=2^3, $2^4 \ldots 2^{12}$. For the decoding case shown in Fig. 1(b), the time of the initialization is ignored, because the main procedure dominates the time consumption when the file size is reasonably large. In the experiment, we record the time spent on repeating the main procedure in 10^4 times. Then we calculate the data throughput by the following formulas:

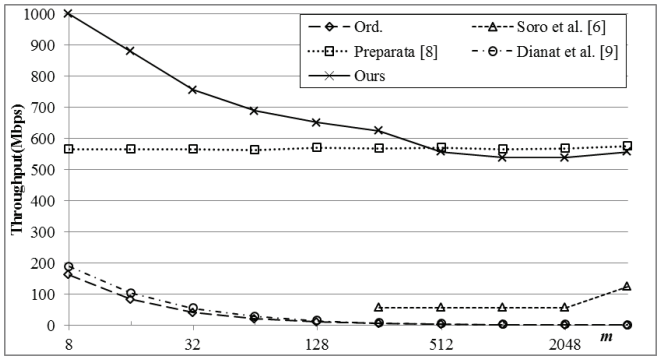
Encoding throughput
= $10^4\times$Size of n codeword symbols/Encoding time;
Decoding throughput
= $10^4\times$Size of k message symbols/Decoding time.

As shown in Fig. 1, for encoding, when $m\leq 128$, the throughput of the proposed algorithm (systematic case) is roughly equal to the [8] (non-systematic case); when m = 8, the throughput of proposed algorithm is about 1.77 times that of the [8]. For decoding, the performance of [6] is very close to the proposed algorithm when code rate is 1/2.

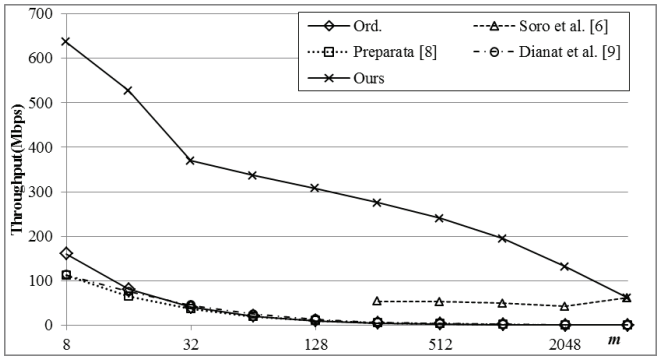
VII. DISCUSSIONS AND CONCLUSIONS

As listed in Table I, the proposed IDA takes fewer operations than the compared FNT-based algorithms. The rationale of improvement is elaborated in the following.

For the systematic case of (n, k) IDA, the encoder reads k input symbols and then generates m parity symbols in each iteration. In encoding, we observe that several FNT-based algorithms employ n-point FNT within $\Theta(n\log n)$. However, for small m, i.e., the high code rate case, the n-point FNT may re-generate the input symbols in the resulting symbols. To remove those redundant overhead, the proposed algorithm employs m-point FNTs to reduce the complexity from $\Theta(n\log n)$ to $\Theta(n\log m)$. In decoding, since the decoder only needs calculating the m erased symbols at most, we can also employ m-point FNT to achieve better performance by following Fermat algorithm.



(a)



(b)

Fig. 1: The real comparisons with ordinary method, [6], [8], [9], and the proposed approach, for n=8192: (a) the encoding case; (b) the decoding case.

REFERENCES

[1] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *J. ACM*, vol. 36, no. 2, pp. 335–348, 1989.

[2] M. O. Rabin, *The Information Dispersal Algorithm and its Applications*. Springer-Verlag, 1990, pp. 406–419.

[3] S. Lin and D. J. Costello, "Decoding of nonbinary BCH and RS codes: the Berlekamp algorithm," in *Error Control Coding*, 2nd edition. Pearson: Prentice-Hall, 2004, ch. 7.4, pp. 241–248.

[4] D. Bini and V. Y. Pan, *Polynomial and Matrix Computations Fundamental Algorithms, Vol. 1*. Birkhäuser, 1994.

[5] F. Didier, "Efficient erasure decoding of Reed-Solomon codes," Computing Research Repository - CORR, vol. abs/0901.1886, 2009.

[6] A. Soro and J. Lacan, "FNT-based Reed-Solomon erasure codes," *2010 IEEE Consumer Communications and Networking Conference*.

[7] 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.

[8] F. P. Preparata, "Holographic dispersal and recovery of information," *IEEE Trans. Inf. theory*, vol. 35, no. 5, Sep. 1989.

[9] R. Dianat and F. Marvasti, "FFT-based fast Reed-Solomon codes with arbitrary block lengths and rates," *IEE Proc. Commun.*, vol. 152, no. 2, pp. 151–156, Apr. 2005.

[10] M. Mitzenmacher, "Digital fountains: a survey and look forward," in *Proc. 2004 IEEE Information Theory Workshop*, pp. 271–276.

[11] J. S. Plank, J. Luo, C. D. Schuman, L. Xu, and Z. Wilcox-O'Hearn, "A performance evaluation and examination of open-source erasure coding libraries for storage," FAST-2009, Feb. 2009.