Computationally-Secure Regenerating Code

Hidenori Kuwakado† Masazumi Kurihara‡

†Kobe University
kuwakado@kobe-u.ac.jp

‡University of Electro-Communications
kuri@ice.uec.ac.jp

Abstract Regenerating codes have been developed for achieving a robust and efficient distributed storage system. Regenerating codes not only reconstruct data but also regenerate a fragment of data stored by a failed storage node using active storage nodes. Secure regenerating codes to protect data against eavesdropping have recently shown. Since secure regenerating codes provide information-theoretic security against eavesdropping, the encoding rate is not sufficiently large. To solve this problem, this paper proposes a computationally-secure regenerating code. The encoding rate of this code asymptotically approaches to that of any non-secure regenerating code, and is independent of security parameters. This paper also proposes a new informationally-secure regenerating code suitable for achieving the computationally-secure regenerating code. The new code provides high security compared with previous informationally-secure codes.

1 Introduction

Internet-based services such as video hosting service and search engine require a large amounts of storage. Since their size is too large to store on one disk and disks may crash, distributed storage systems are absolutely essential for such Internet-based services. Distributed storage systems provide reliable access to data on individually unreliable nodes. A typical manner for achieving a reliable distributed storage is to store data across n nodes in such a way that each node stores a fragment of data (*share*) in size α and assembling k ($< n$) shares allows us to reconstruct data. Due to this mechanism, even if one of nodes fails, we have no need to worry about losing data. However, we have to recover the failed node to maintain the distributed storage system. In particular, it is desirable to regenerate the share of the failed node efficiently as possible.

Dimakis et al. [1] have formalized not only the *reconstruction* of data but also the *regeneration* of the share when a node fails (Fig. 1). They proposed the concept of regenerating codes such that the share of the failed node can be regenerated by any d nodes. We call data for regenerating the share a *piece*, which a node computes from a share by itself. The size of a piece is denoted by β. The major concern of regenerating codes is two quantities: the size of k shares for reconstructing data (i.e., αk), the size of d pieces for regenerating a share (i.e., βd). It is desirable that the two quantities are small as possible, but Dimakis et al. showed that the two quantities are traded and βd can be smaller than αk. The later fact means that the share can be regenerated without reconstructing data. The value of βd is usually called the *bandwidth*. The regenerating code such that the bandwidth is minimized is called a *minimum bandwidth regenerating code* (an MBR code). In contrast, the regenerating code such that the share size per node is minimized is called a *minimum storage regenerating code* (an MSR code). The systematic construction of MBR/MSR codes was given in articles [10, 12].

Regenerating codes may be similar to information dispersal algorithms [9] and secret sharing schemes [7]. The differences between them are summarized below. The information dispersal algorithm was developed to provide reliable transmission of data in distributed networks. In the context of the information dispersal algorithm, reconstructing data from k out of n shares is of concern, but regenerating the share of a failed node is not covered by the information dispersal algorithm. The secret sharing scheme produces shares in such a way that a share does not give any information about data. The share produced with the regenerating code does not have such a property usually. In the context of the secret sharing scheme as well as the information dispersal algorithm, regenerating the share of the failed node is not covered by the secret sharing scheme.

Combining a regenerating code with a secret sharing scheme has recently been proposed in articles [4, 5, 6, 8]. Such a code is called an *informationally-secure regenerating code* (an IS-R code). In addition to properties of the regenerating code, the IS-R code provides the security such that even if an eavesdropper knows some shares (or pieces), no information about data leaks to the eavesdropper. In order to achieve

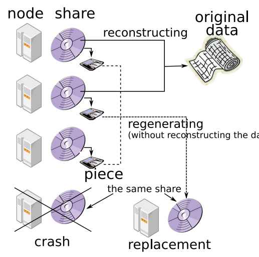

Figure 1: Concept of regenerating codes.

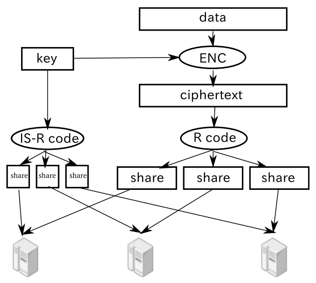

Figure 2: Proposed CS-R code.

the informational security, the IS-R code requires many random symbols. Hence, a previous IS-R code with a high security provides a low encoding rate.

To improve the low encoding rate, this paper proposes a *computationally-secure regenerating code* (a CS-R code). The proposed CS-R code consists of a computationally-secure encryption, an IS-R code, and a non-secure regenerating code (Fig. 2). The encoding rate of the proposed CS-R code asymptotically approaches to the underlying non-secure regenerating code as the size of data becomes large. The proposed CS-R code requires the IS-R code that provides high security with respect to k. Since previous IS-R codes cannot satisfy the requirement, this paper also proposes a new IS-R code.

2 Computationally-Secure Regenerating Code

2.1 Definition

We start by defining a regenerating code [1]. Let M be a random variable representing a message (data) to be distributed. The message M is encoded to n *shares* $S_1, \ldots, S_n$ in such a way that M can be successfully reconstructed from any k out of n shares. This reconstruction property can be written as follows: for random variables $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ representing any k shares,

$$H(M|S_{i_1}, S_{i_2}, \ldots, S_{i_k}) = 0, \tag{1}$$

where H is the Shannon entropy and symbols are random variables representing them. Suppose that a node z fails. The failed node z (or a replacement node) tries to regenerate the share S_z by connecting any d active nodes. The active node i computes a *piece* $P_{z,i}$ from the share S_i. Note that $H(P_{z,i}|S_i, z) = 0$, that is, $P_{z,i}$ is uniquely determined from S_i and z. The active node i gives the piece $P_{z,i}$ to the failed node z. This regeneration property can be written as

$$H(S_z|P_{z,i_1}, P_{z,i_2}, \ldots, P_{z,i_d}) = 0. \tag{2}$$

In articles on regenerating codes, the share is called a stored message or stored data, and the piece is often called downloaded data. Let α be the size of a share per node, let β be that of a piece per node, and let B denote that of a message. The size is measured with the number of elements in a finite field over which the encoding, the reconstructing, and the regenerating operation are performed. The regenerating code is associated with the collection of parameters $[n, k, d, \alpha, \beta, B]$.

We call a regenerating code an *informationally-secure regenerating code* (IS-R code) if the regenerating code satisfies the following conditions, which refine the definition of secrecy capacity [8]. Let M be a random variable that is uniformly distributed, representing a secrete message.

- Let S_i be a random variable representing the share of a node i. For any τ $(\le \ell_c)$ random variables $S_{i_1}, S_{i_2}, \ldots, S_{i_\tau}$,

$$H(M|S_{i_1}, S_{i_2}, \ldots, S_{i_\tau}) = H(M) \quad (1 \le \tau \le \ell_c). \tag{3}$$

  Note that ℓ_c is implicitly less than k because of Eq. (1).

- Let $P_{z,i}$ be a random variable representing the piece that a node i produces for regenerating the share of a failed node z. For any z and any τ $(\le \ell_g)$ random variables $P_{z,i_1}, P_{z,i_2}, \ldots, P_{z,i_\tau}$,

$$H(M|P_{z,i_1}, P_{z,i_2}, \ldots, P_{z,i_\tau}) = H(M) \quad (1 \le \tau \le \ell_g). \tag{4}$$

Secret capacity is subject only to Eq. (1) and Eq. (3). Since an eavesdropper can obtain pieces $P_{z,i}$ when a node z fails, we include Eq. (4) in the definition. Since $P_{z,i}$ is uniquely determined from S_i and z, we have $\ell_c \leq \ell_g$. The IS-R code is associated with the collection of parameters $[\ell_c, \ell_g]$ that represents resistance to eavesdropping.

We now proceed to define a *computationally-secure regenerating code* (CS-R code). Let κ be a security parameter. Polynomial-time algorithms in κ for producing n shares S_i and producing pieces $P_{z,i}$ are denoted by Encode_s and Encode_p, respectively. The reconstruction property and the regeneration one are defined in the same way as Eq. (1) and Eq. (2), respectively. If the following conditions in addition to them are satisfied, the regenerating code is called a computationally-secure regenerating code. Let $T = M^{(1)} \parallel M^{(2)} \parallel Z$ be a random variable representing two messages $M^{(1)}, M^{(2)}$ and additional information Z that may depend on $M^{(1)}, M^{(2)}$ (e.g., $Z = M^{(1)} \parallel M^{(2)}$).

- Let $S_i^{(j)}$ be a random variable representing the share of a node i for $M^{(j)}$. For every polynomial-time (in κ) algorithm A, every positive polynomial poly(κ), every $\tau \leq \ell_c$, and all sufficiently large κ,

$$\left|\Pr\left[A(Z, S_{i_1}^{(1)}, S_{i_2}^{(1)}, \ldots, S_{i_\tau}^{(1)}) = 1\right] - \Pr\left[A(Z, S_{i_1}^{(2)}, S_{i_2}^{(2)}, \ldots, S_{i_\tau}^{(2)}) = 1\right]\right| < \frac{1}{\mathrm{poly}(\kappa)},$$

  where the probability in the above terms is taken over the probability space underlying T and the internal coin toss of Encode_s and A. The above inequality means that it is hard to distinguish between the space of τ shares corresponding to $M^{(1)}$ and that corresponding to $M^{(2)}$.

- Let $P_{z,i}^{(j)}$ be a random variable representing the piece that a node i produces for regenerating the share of a failed node z for $M^{(j)}$. For every polynomial-time (in κ) algorithm A, every positive polynomial poly(κ), every $\tau \leq \ell_g$, any τ of n pieces, and all sufficiently large κ,

$$\left|\Pr\left[A(Z, P_{z,i_1}^{(1)}, P_{z,i_2}^{(1)}, \ldots, P_{z,i_\tau}^{(1)}) = 1\right] - \Pr\left[A(Z, P_{z,i_1}^{(2)}, P_{z,i_2}^{(2)}, \ldots, P_{z,i_\tau}^{(2)}) = 1\right]\right| < \frac{1}{\mathrm{poly}(\kappa)},$$

  where the probability in the above terms is taken over the probability space underlying T and the internal coin toss of $\mathsf{Encode}_s, \mathsf{Encode}_p$, and A.

The CS-R code is associated with the collection of parameters $[\kappa, \ell_c, \ell_g]$ that represents resistance to eavesdropping.

2.2 Construction

Figure 2 illustrates the proposed CS-R code. In Fig. 2, 'ENC', 'R code', and 'IS-R code' denote a computationally-secure encryption, a non-secure regenerating code, an IS-R code, respectively. Parameters $[n, k, d]$ of the non-secure regenerating code must be equal to those of the IS-R code. Any encryption scheme, any non-secure regenerating code, and any IS-R code are available because they are used as black boxes. This construction is similar to a computationally-secure secret sharing scheme that consists of a computationally-secure encryption, an information dispersal algorithm, and a secret sharing scheme [3]. We intuitively understand that this code is the CS-R code. Since the key is encoded with the IS-R code, the key is unknown unless k shares are collected. When the key is unknown, the ciphertext does not give any information about the message in the sense of computational security. The proof is available from authors.

The proposed CS-R code is associated with parameters $[n, k, d, \alpha, \beta]$ and $[\kappa, \ell_c, \ell_g]$. When the size of data is much larger than that of the key, the share size α and the piece size β are primarily characterized by the underlying regenerating code. In contrast, the security of data depends on the encryption and the IS-R code. The key size κ is equal to that of the encryption scheme, parameters ℓ_c, ℓ_g are equal to those of the IS-R code. The next section shows a new IS-R code that provides high security, that is, $\ell_c = k-1, \ell_g = d-1$.

3 New Informationally-Secure Regenerating Code

3.1 Encoding, Reconstructing, and Regenerating

We propose a new IS-R code that is associated with the collection of parameters $[n, k, d, \alpha, \beta] = [n, k, 2k-2, k-1, 1]$ and $[\ell_c, \ell_g] = [k-1, 2k-3]$. Note that n and β are independent of k and other parameters depend on k. The IS-R code with this parameters is important for two reasons: (1) data is protected safely with respect to the parameter k, (2) No IS-R code with such a parameter has been shown.

Encoding Suppose that the key is $k-1$ symbols that are chosen from a finite field $\mathbb{F}_q$ uniformly and independently. Let $\check{k} = k(k-1)/2$. Consider the following $2(k-1) \times (k-1)$ matrix U that consists of two $(k-1) \times (k-1)$ symmetric matrices U_T, U_B.

$$U = \begin{pmatrix} U_T \\ U_B \end{pmatrix} = \begin{pmatrix} u_1 & u_2 & u_3 & \cdots & u_{k-1} \\ u_2 & u_k & u_{k+1} & \cdots & u_{2k-3} \\ u_3 & u_{k+1} & u_{2k-2} & \cdots & u_{3k-6} \\ & & \vdots & & \\ u_{k-1} & u_{2k-3} & u_{3k-6} & \cdots & u_{k(k-1)/2} \\ & & & & \\ u_{\check{k}+1} & u_{\check{k}+2} & u_{\check{k}+3} & \cdots & u_{\check{k}+k-1} \\ u_{\check{k}+2} & u_{\check{k}+k} & u_{\check{k}+k+1} & \cdots & u_{\check{k}+2k-3} \\ u_{\check{k}+3} & u_{\check{k}+k+1} & u_{\check{k}+2k-2} & \cdots & u_{\check{k}+3k-6} \\ & & \vdots & & \\ u_{\check{k}+k-1} & u_{\check{k}+2k-3} & u_{\check{k}+3k-6} & \cdots & u_{\check{k}+k(k-1)/2} \end{pmatrix}$$

where the first $k-1$ elements $(u_1, u_2, \ldots, u_{k-1})$ are the key and each other element is chosen from $\mathbb{F}_q$ uniformly and independently. The matrix U essentially consists of $k-1$ key elements and $(k-1)^2$ random elements. The structure of U is similar to that of the message matrix of the MSR code proposed by Rashmi, Shah, and Kumar (called the *RSK-MSR code*) [10]. Assign a unique and public symbol x_i in $\mathbb{F}_q$ to a node i in such a way that the following three conditions are satisfied.

Condition of x_i:
1. For any i, j, $x_i^{k-1} \neq x_j^{k-1}$. (for the reconstruction and the regeneration)
2. The matrix Υ defined by Eq. (6) is non-singular. (for the security of Eq. (3))
3. For any i, $x_i \neq 0$. (for the security of Eq. (3) and Eq. (4))

The share of node i, denoted $\underline{c}_i$, is computed as

$$\underline{c}_i^t = \begin{pmatrix} c_{i,1} & c_{i,2} & \ldots & c_{i,k-1} \end{pmatrix} = (1, x_i, x_i^2, \ldots, x_i^{2k-3})U = \begin{pmatrix} \underline{\phi}_i^t & x_i^{k-1} \cdot \underline{\phi}_i^t \end{pmatrix} U \tag{5}$$

where the superscript t denotes its transportation, $\underline{\phi}_i = (1, x_i, \ldots, x_i^{k-2})^t$, and all the operations are done over $\mathbb{F}_q$. The complexity of encoding is $O(k^2)$.

The share size α, which is the number of elements in $\underline{c}_i$, is $k-1$. Let us compare the share size with that of a k-out-of-n threshold secret sharing scheme. Eq. (1) and Eq. (3) are properties that a k-out-of-n threshold secret sharing scheme has to satisfy. The share size of any k-out-of-n threshold secret sharing scheme is not smaller than the size of original data, and a secret sharing scheme is said to be *ideal* if the share size is equal to the data size. Accordingly, the proposed IS-R code is the ideal k-out-of-n threshold secret sharing scheme. Note that IS-R codes in articles [4, 5] are not k-out-of-n threshold secret sharing scheme, and IS-R codes in articles [6, 8] are not ideal.

We leave a generic method for the second condition of x_i as an open problem. The determinant of Υ of Eq. (6) for k, denoted $\det(\Upsilon_k)$, depends on the value of k. Examples of $\det(\Upsilon_k)$ are given below. Symbols $x_1, x_2, \ldots$ instead of generic symbols $x_{i_1}, x_{i_2}, \ldots$ are used.

$$\det(\Upsilon_2) = x_1, \qquad \det(\Upsilon_3) = x_1^3 x_2^3 (x_1 - x_2)\left(x_2^2 - x_1^2\right),$$

$$\det(\Upsilon_4) = \prod_{i=1}^{3} x_i^5 \prod_{1 \le i < j \le 3} (x_i - x_j)^2 (x_j^3 - x_i^3), \qquad \det(\Upsilon_5) = \prod_{i=1}^{4} x_i^7 \prod_{1 \le i < j \le 4} (x_i - x_j)^3 (x_j^4 - x_i^4).$$

Since Υ is a $(k-1)^2 \times (k-1)^2$ matrix, the complexity for computing $\det(\Upsilon_k)$ is $O(k^6)$. We were not able to obtain a general formula for $\det(\Upsilon_k)$. In the above cases, the first and the third condition of x_i are included by the second condition.

Reconstruction and Regeneration The reconstruction of the key is similar to the reconstruction of the RSK-MSR code. Unlike the RSK-MSR code, it is sufficient to reconstruct only the first row of U (i.e, key elements). The regeneration of a share is the same as that of the RSK-MSR code. The complexity of reconstructing and that of regenerating are $O(k^3)$.

3.2 Security

We first prove the security condition of Eq. (3), that is, $H(M|S_{i_1}, \dots, S_{i_{k-1}}) = H(M)$ where S_{i_j} is a random variable representing the share of a node i_j, denoted $\underline{c}_{i_j}$. For simplifying the notation, let $(i_1, i_2, \dots, i_{k-1}) = (1, 2, \dots, k-1)$. Since $\underline{c}_i$ is computed by Eq. (5), we have

$$\begin{pmatrix} \underline{c}_1^t \\ \underline{c}_2^t \\ \vdots \\ \underline{c}_{k-1}^t \end{pmatrix} = \begin{pmatrix} \underline{\phi}_1^t & x_1^{k-1} \cdot \underline{\phi}_1^t \\ \underline{\phi}_2^t & x_2^{k-1} \cdot \underline{\phi}_2^t \\ \vdots & \\ \underline{\phi}_{k-1}^t & x_{k-1}^{k-1} \cdot \underline{\phi}_{k-1}^t \end{pmatrix} U,$$

We can transform the above equation into

$$\begin{pmatrix} c_{1,1} - g_{1,1} \\ \vdots \\ c_{1,k-1} - g_{1,k-1} \\ \vdots \\ c_{k-1,1} - g_{k-1,1} \\ \vdots \\ c_{k-1,k-1} - g_{k-1,k-1} \end{pmatrix} = \Upsilon \begin{pmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k(k-1)} \end{pmatrix}, \tag{6}$$

where $g_{i,j}$ is a polynomial that includes only key symbols $u_1, u_2, \dots, u_{k-1}$ and x_i^j, and all the elements in the matrix Υ are in $\{0, x_i^j\}$, that is, the matrix Υ does not include any u_i. Recall the second assumption on the choice of x_i: each x_i is chosen in such a way that Υ is non-singular. If $(u_k, u_{k+1}, \dots, u_{k(k-1)})$ is uniformly chosen from $\mathbb{F}_q^{(k-1)^2}$, then $(c_{1,1}, \dots, c_{k-1,k-1})$ is uniformly distributed over $\mathbb{F}_q^{(k-1)^2}$. This completes the proof.

We next prove the security condition of Eq. (4), that is, $H(M|P_{z,i_1}, P_{z,i_2}, \dots, P_{z,i_{2k-3}}) = H(M)$. For simplifying the notation, let $(i_1, i_2, \dots, i_{2k-3}) = (1, 2, \dots, 2k-3)$. The piece $P_{z,i}$ is denoted by $d_{z,i}$. The following system of equations is obtained from $2k-3$ pieces $d_{z,i}$.

$$\begin{pmatrix} d_{z,1} \\ d_{z,2} \\ \vdots \\ d_{z,2k-3} \end{pmatrix} = \begin{pmatrix} \underline{\phi}_1^t & x_1^{k-1} \underline{\phi}_1^t \\ \underline{\phi}_2^t & x_2^{k-1} \underline{\phi}_2^t \\ \vdots & \vdots \\ \underline{\phi}_{2k-3}^t & x_{2k-3}^{k-1} \underline{\phi}_{2k-3}^t \end{pmatrix} U \underline{\phi}_z = \begin{pmatrix} 1 & x_1 & \dots & x_1^{k-2} & x_1^{k-1} & \dots x_1^{2k-3} \\ 1 & x_2 & \dots & x_2^{k-2} & x_2^{k-1} & \dots x_2^{2k-3} \\ \vdots & & & & & \\ 1 & x_{2k-3} & \dots & x_{2k-3}^{k-2} & x_{2k-3}^{k-1} & \dots x_{2k-3}^{2k-3} \end{pmatrix} \underline{\psi}_z, \tag{7}$$

where $\underline{\psi}_z = U \underline{\phi}_z = (\psi_{z,1}, \psi_{z,2}, \dots, \psi_{z,2k-2})^t$. The key is related only to $\psi_{z,1}, \psi_{z,2}, \dots \psi_{z,k-1}$ because key symbols locate on the first row and the first column of U_T. Since each diagonal element in U_T below the second row is random, each $\psi_{z,i}$ for $2 \le i \le k-1$ is distributed on $\mathbb{F}_q$ uniformly and independently. Each $\psi_{z,i}$ for $k \le i \le 2k-3$ is independent of the key, and is distributed on $\mathbb{F}_q$ uniformly and independently. Equation (7) is transformed into

$$\begin{pmatrix} d_{z,1} - \psi_{z,1} \\ d_{z,2} - \psi_{z,1} \\ \vdots \\ d_{z,2k-3} - \psi_{z,1} \end{pmatrix} = \begin{pmatrix} x_1 & \dots & x_1^{k-2} & x_1^{k-1} & \dots x_1^{2k-3} \\ x_2 & \dots & x_2^{k-2} & x_2^{k-1} & \dots x_2^{2k-3} \\ \vdots & & & & \\ x_{2k-3} & \dots & x_{2k-3}^{k-2} & x_{2k-3}^{k-1} & \dots x_{2k-3}^{2k-3} \end{pmatrix} \begin{pmatrix} \psi_{z,2} \\ \psi_{z,3} \\ \vdots \\ \psi_{z,2k-2} \end{pmatrix} = \tilde{V} \begin{pmatrix} \psi_{z,2} \\ \psi_{z,3} \\ \vdots \\ \psi_{z,2k-2} \end{pmatrix}.$$

The matrix $\tilde{V}$ is non-singular when the conditions of x_i are satisfied. Hence, since each $\psi_{z,i}$ $(i \ge 2)$ is distributed on $\mathbb{F}_q$ uniformly and independently, each piece $d_{z,i}$ is also distributed on $\mathbb{F}_q$ uniformly and independently. This completes the proof.

Table 1: Achievable parameters for k ($\beta = 1$).

Code	R	d	α	ℓ_c	ℓ_g	
article [4]	$(k-2)/k$	$2(k-1)$	$k-1$	2	d	
article [5]	$(\alpha-1)/(\alpha+1)$	k	α	k/α	d	k is a multiple of α.
article [6]	$2/(k(k+1))$	k	k	$k-1$	$d-1$	ℓ_c is maximized.
article [8]	$(d+1-k)/\sum_{i=1}^{k}(d+1-k)$	d	$d-1$	$k-1$	d	ℓ_c is maximized. $n = d-1$.
this paper	$1/k$	$2(k-1)$	$k-1$	$k-1$	$d-1$	

Table 2: Throughput of the reconstruction. [cycles/bit]

$\kappa \setminus k$	3	5	9	17	$\mathbb{F}_q$
128	1,010	4,351	23,491	188,778	GF(2^8)
256	891	2,188	14,674	105,134	GF(2^{16})
512	833	2,009	7,343	64,238	GF(2^{32})
1024	985	2,176	7,152	35,939	GF(2^{64})

Table 3: Throughput of the regeneration. [cycles/bit]

$\kappa \setminus d$	4	8	16	32	$\mathbb{F}_q$
128	813	3,555	13,614	70,847	GF(2^8)
256	705	1,622	8,347	38,701	GF(2^{16})
512	717	1,534	4,353	22,633	GF(2^{32})
1024	931	1,800	4,488	14,142	GF(2^{64})

4 Comparison and Implementation

We define the encoding rate R of an IS-R code as $R = N_d/(N_d + N_r)$ where N_d is the size of data and N_r is the size of random symbols. The size is measured with the number of elements in $\mathbb{F}_q$. Table 1 shows achievable parameters of IS-R codes for a given k. We see that the proposed IS-R code is as secure as previous codes in terms of $[\ell_c, \ell_g]$. Compared with the IS-R codes proposed in articles [6, 8], the encoding rate is improved by a factor of k.

We implemented the proposed IS-R code using NTL [11] on the 32-bit Linux (CPU: Intel Celeron 560, 2.13 [GHz]). Table 2 and Table 3 show the throughput of the reconstruction and that of the regeneration, respectively. In these tables, κ denotes the bit length of data (a key) and the throughput is the number of CPU cycles for the reconstruction (or the regeneration) per one bit of data (or share). For example, when $\kappa = 128, k = 17$, the reconstruction and the regeneration are completed in 0.012 [s] and 0.0045 [s], respectively. The complexity of throughput for small k seems to be less than $O(k^3)$, which is the time complexity of the reconstruction and that of the regeneration. As κ is large, the throughput is improved in almost all cases. This suggests that the overhead of operations over the finite field is not negligible.

References

[1] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," IEEE Transactions on Information Theory, vol. 56, no. 9, pp. 4539–4551, 2010.

[2] O. Goldreich, "A uniform-complexity treatment of encryption and zero-knowledge," Journal of Cryptography, vol. 6, no. 1, pp. 21–53, 1993.

[3] H. Krawczyk, "Secret sharing made short," Advances in Cryptology - CRYPTO '93, Lecture Notes in Computer Science, vol. 773, pp. 136–146, 1993.

[4] M. Kurihara and H. Kuwakado, "On regenerating codes and secret sharing for distributed storage," IEICE Technical Report, vol. IEICE-110, no. IEICE-IT-363, pp. 13–18, 2011.

[5] M. Kurihara and H. Kuwakado, "On an extended version of Rashmi-Shah-Kumar regenerating codes and secret sharing for distributed storage," IEICE Technical Report, vol. IEICE-110, no. IEICE-IT-442, pp. 303–310, 2011.

[6] M. Kurihara and H. Kuwakado, "On ramp secret sharing schemes for distributed storage systems under repair dynamics," IEICE Technical Report, vol. IEICE-111, no. IEICE-IT-142, pp. 41–46, 2011.

[7] R. J. McEliece and D. V. Sarwate, "On sharing secrets and Reed-Solomon codes," Communications of the ACM, vol. 24, no. 9, pp. 583–584, 1981.

[8] S. Pawar, S. Y. E. Rouayheb, and K. Ramchandran, "On secure distributed data storage under repair dynamics," `http://arxiv.org/abs/1003.0488`, 2010.

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

[10] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," `http://arxiv.org/abs/1005.4178`, 2010.

[11] V. Shoup, "NTL: A library for doing number theory," `http://shoup.net/ntl/`, 2005.

[12] C. Suh and K. Ramchandran, "Exact regeneration codes for distributed storage repair using interference alignment," `http://arxiv.org/abs/1001.0107`, 2010.