An Empirical Study on Data Retrievability in Decentralized Erasure Code Based Distributed Storage Systems

Hsiao-Ying Lin, Li-Ping Tung, Bao-Shuh P. Lin
Intelligent Information and Communications Research Center
National Chiao Tung University
hsiaoying.lin@gmail.com, lptung@nctu.edu.tw, bplin@mail.nctu.edu.tw

***Abstract*—Erasure codes are applied in distributed storage systems to provide data robustness against server failures by storing data redundancy among many storage servers. A (n, k) erasure code encodes a data object, which is represented as k elements, into a codeword of n elements such that any k out of these n codeword elements can recover the data object back. Decentralized erasure codes are proposed for distributed storage systems without a central authority. The characteristic of decentralization makes resulting storage systems more scalable and suitable for loosely-organized networking environments. However, different from conventional erasure codes, decentralized erasure codes trade some probability of a successful data retrieval for decentralization. Although theoretical lower bounds on the probability are overwhelming from a theoretical aspect, it is essential to know what the data retrievability is in real applications from a practical aspect. We focus on decentralized erasure code based storage systems and investigate data retrievability from both theoretical and practical aspects. We conduct simulation for random processes of storage systems to evaluate data retrievability. Then we compare simulation results and analytical values from theoretical bounds. By our comparison, we find that data retrievability is underestimated by those bounds. Data retrievability is over 99% in most cases in our simulations, where the order of the used finite field is an 8-bit prime. Data retrievability can be enlarged by using a larger finite field. We believe that data retrievability of decentralized erasure code based storage systems is acceptable for real applications.**

Keywords*—*erasure codes; code based distributed storage systems; fault tolerance; data retrievability

I. Introduction

To provide data robustness against server failures, many coding techniques are applied on distributed storage systems. Erasure codes provide data robustness by storing data redundancy among many storage servers. A (n, k) erasure code encodes a data object, which is decomposed as k elements, into a codeword of n elements such that any k out of these n codeword elements can recover the data object back. When a storage server stores a codeword element, the storage system consisting of n storage servers tolerates $(n - k)$ failed storage servers.

The concept of network coding is also applied in distributed networked storage systems [1], [2], [3], where each storage server can encode received data and then store encoded data. A data collector can recover the original data object back by accessing k storage servers. The distributed storage system is modeled as the information flow graph as shown in Fig. 1.

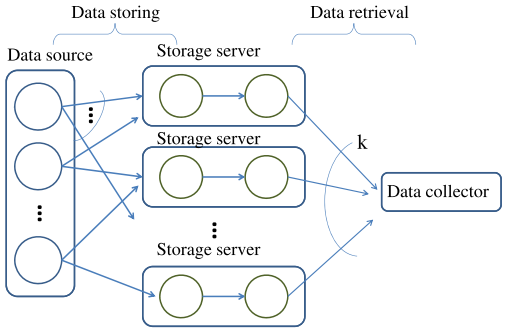


Fig. 1. The information flow graph of a distributed storage system.

A node models an element of a data object, a receiver or sender interfaces of a storage server, or a data collector. An edge from an element to a storage server models data storing. An internal edge in a storage server models the storage cost of the server. An edge from a server to the data collector models data retrieval. A successful data retrieval is possible if the flow of the min-cut is at least the size of the data object. These codes are sophisticatedly designed such that any possible data collector can successfully retrieve the data object. Data retrievability is evaluated as the probability of a successful data retrieval over all possible data collectors.

For storage systems without a central authority, decentralized erasure codes are proposed. The characteristic of decentralization makes resulting storage systems more scalable and suitable for loosely-organized networking environments. However, different from conventional erasure codes, decentralized erasure codes trade some data retrievability for decentralization. That is, few subsets of k storage servers cannot recover the original data object.

In light of the beauty of decentralization, secure decentralized erasure codes are proposed by integrating decentralized erasure codes and data encryption schemes to simultaneously address data robustness and confidentiality in storage systems [4], [5]. A new role of key servers is added into the proposed storage system to provide key management services. The newly proposed system model is called secure system model in this paper. In this secure storage system, data retrievability is at least $1 - k/p - \epsilon(k)$, where p is the order of a finite field and $\epsilon(k) = o(1)$ approaches 0 as k gets larger.

A specific distributed repair mechanism is designed for de-

978-0-7695-5021-3/13 $26.00 © 2013 IEEE
DOI 10.1109/SERE.2013.27

centralized erasure code based storage systems to maintain data redundancy [6]. When the storage system with failed storage servers still has data retrievability at least $1 - k/p - \epsilon(k)$, the distributed repair mechanism repairs all failed storage servers such that the repaired storage system has no failed storage server and data retrievability at least $1 - 2k/p - e^{-k} - \epsilon(k)$.

Previous studies analyze and lower bound the probability of a successful data retrieval taken over all possible k-subsets of n storage servers. Although these bounds are overwhelming from a theoretical aspect, it is essential to know what data retrievability is in real applications from a practical aspect. Our work is the first attempt to evaluate data retrievability of decentralized erasure code based storage systems in real applications and investigate the gap between analytical values and simulation results.

We focus on the family of decentralized erasure code based storage systems in two different system models with the distributed repair mechanism. The two system models have common bounds on data retrievability but differ in the way of data retrieval. The first system model has a single data collector, who directly queries storage servers to recover the data object. The second one has distributed key servers, who independently query storage servers, and a data collector, who collects data from key servers to recover the data object back. We would go into details about these two system models in Section III.

Our main goal is to know how data retrievability would be in real applications and investigate the gap between analytical values and simulation results. Specifically, we are interested in the following issues:

- For a storage system in an initial state (i.e., without corruption), we wants to know what data retrievability is. We also investigate the gap between simulation results and the theoretical bound $1 - k/p - \epsilon(k)$.
- For the distributed repair mechanism, we want to know under what conditions a storage system can keep satisfying the assumption on data retrievability (i.e. at least $1 - k/p - \epsilon(k)$). The results would provide a reference for a system manager to decide when to execute the repair mechanism.
- We are also interested in under what conditions a repaired storage system can satisfy the theoretical bound $1 - 2k/p - e^{-k} - \epsilon(k)$.
- By comparing the previous two results, we can observer that whether the assumption and the bound on data retrievability are tight.

We derive analytical values from theoretical bounds and evaluate the probability of a successful data retrieval by Monte-Carlo simulations with $10,000$ trials. We simulate random processes of data storing, storage server failures, data retrieval, and the distributed repair mechanism. The probability of a successful data retrieval is evaluated in three system states: the initial state, the state with failure and the repaired state. We also compare simulation results with analytical values derived from theoretical bounds.

By our results, we make the following observations.

- Simulation results show that data retrievability is much underestimated by theoretical bounds for both initial systems and repaired systems. For initial systems, in most cases of parameter settings in our study, data retrievability is over 99%. For repaired systems, when the fraction of failed storage servers is less than 30%, data retrievability is over 95%.
- When the fraction of surviving storage servers keeps over some threshold, the assumption on data retrievability of the distributed repair mechanism holds. Among all cases in our study, the highest threshold is 76%. Our results characterize conditions for executing the distributed repair mechanism.
- When a storage system satisfies the assumption, the repaired storage system has a better data retrievability than the theoretical bound. In other words, the assumption is overestimated on required data retrievability.

Our study explores data retrievability of decentralized erasure code based storage systems in the initial state, the state with failure, and the repaired state. Our simulations are operated in a finite field of a 8-bit prime order. By simulation results, the probability of a successful data retrieval is underestimated by theoretical bounds. By using a larger finite field, data retrievability will be higher. We believe that data retrievability is acceptable for most real applications which usually use a large finite field. Our results are positive evidences showing that decentralized erasure code based storage systems are practical in terms of data retrievability.

The rest of this paper is organized as follows. We briefly review code based storage systems and repair mechanisms in Section II. Decentralized erasure code based storage systems with the distributed repair mechanism are introduced in Section III. Simulations and results are presented in Section IV. The conclusion are drawn in Section V.

II. Related Work

We briefly review code based distributed storage systems and repair mechanisms.

Code based distributed storage systems store encoded data over many storage servers to provide data robustness against server failures. As we mention in Section I, these codes are designed such that in a storage system of n storage servers, any k out of n storage servers can recover the data object. A (n, k) linear code is an example.

The concept of network coding is also applied in distributed storage systems [1], [2], [3]. Each storage server not only receives data but also performs computation on received data and stores the results. In this approach, a storage system is modeled as an information flow graph. It has been proved that when the flow of the min-cut in the graph is at least the size of the data object, a code exists for the storage system. Hence, the probability of a successful data retrieval is 100%.

Dimakis et el. [7] propose decentralized erasure code based distributed storage systems. The probability of a successful data retrieval is taken over all possible k-subsets of n storage servers. By theoretical studies, the probability is lower bounded by $1 - k/p - o(1)$ under certain system settings, where p is

the order of the operated finite field and $o(1)$ approaches 0 as k gets larger. Later, Lin and Tzeng [4], [5] propose secure decentralized erasure code based distributed storage systems. They present a new system model, which consists of storage servers and key servers, and integrate decentralized erasure codes and data encryption schemes to address data robustness, data confidentiality and secure data forwarding. They provide a theoretical bound on data retrievability under various system parameter settings.

Repair mechanisms are means of data redundancy maintenance. When storage servers fail over time, new storage servers are added into the storage system to replace failed ones. Surviving storage servers are then called old or available storage servers. Repair mechanisms define how new storage servers collect data from old storage servers such that after data collection, any k of n storage servers, no matter new or old, can recover the original data object.

Regenerating codes are proposed with objectives to minimizing repair bandwidth, the amount of transmitted bits for repairing, storage cost, and the amount of stored data per storage server [1]. The tradeoff between repair bandwidth and storage cost is characterized as a curve. One of important results of regenerating codes is that by collecting data from $n-1$ old storage servers, a new storage server utilizes minimum bandwidth for repairing a failed storage server. More constructions and discussions of regenerating codes can be found in [8], [9], [10], [11], [12], [13]. These methods repair storage systems from single-server failure.

Later, many repair mechanisms against multiple-server failures are proposed [14], [15], [16], [17], [18], [19], [20], [6]. Hu et al. propose a mutually cooperative recovery mechanism [15], where new storage servers not only communicate with old storage servers but also communicate with each other to recover the storage system. Therefore, a new storage server has to query $n-1$ storage servers. Wang et al. propose a multi-loss flexible recovery mechanism [14], where a new storage server queries a different number of old storage servers and the objective is to optimize the overall bandwidth for repairing all failed storage servers. Kenneth propose cooperative regenerating codes [16], where a new storage server gets different amount of data from a new storage server and an old storage server. The tradeoff between storage cost and repair bandwidth is also given. Above mechanisms [14], [15], [16], [18], [19], [20] are analyzed by using the min-cut bound of the information flow graph.

Oggier and Datta [17] propose self-repairing homomorphic codes, where each new storage server queries a specific set of old storage servers to regenerate a specific codeword element. A central table is needed for storing the mapping from codeword elements to their specific subsets of old storage servers. Hence, this method is not suitable for decentralized erasure code based storage systems.

Lin et al. propose a distributed repair mechanism for decentralized erasure code based distributed storage systems [6]. The repair mechanism repairs multiple failed storage servers at the same time and minimizes the number of old storage servers a new one has to query while keeping the probability of a successful data retrieval overwhelming. Consider a storage system contains multiple failed storage servers. The assumption on the system is that data retrievability is at least $1-k/p-\epsilon(k)$, where $\epsilon(k) = o(1)$. After being repaired, the system has the same number of storage servers without failures and data retrievability is lower bounded by $1-2k/p-e^{-k}-\epsilon(k)$.

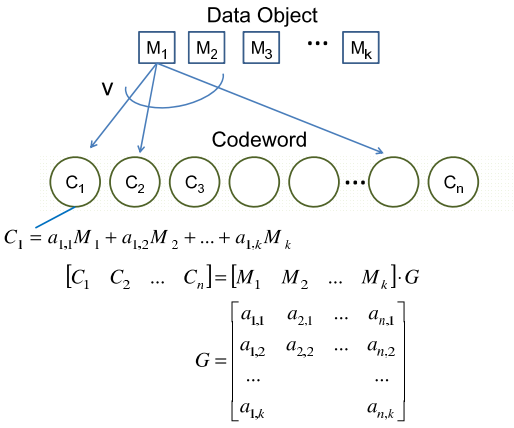


Fig. 2. The encoding processes of decentralized erasure codes.

Data retrievability in above mentioned studies is all theoretically analyzed. To our best knowledge, our work is the first attempt to investigate the probability of a successful data retrieval by Monte-Carlo simulations.

III. Background

We briefly introduce decentralized erasure code based storage systems and their two different system models. Then we review the distributed repair mechanism for these systems. Theoretical bounds on data retrievability under certain parameter settings are presented for initial systems and repaired systems, respectively.

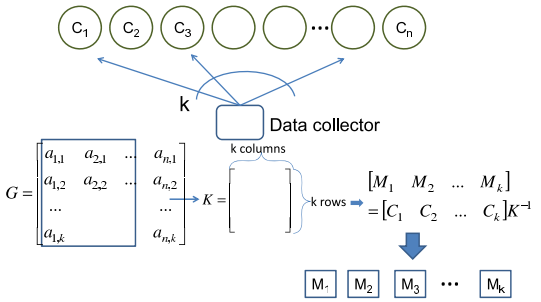
A. Decentralized Erasure Code Based Storage Systems

We consider two system models. They have the same encoding process and differ in the way of data retrieval. To better distinguish them, we call the first one as basic system model. The second one is called secure system model since it is proposed for a storage system supporting data confidentiality.

The encoding process of decentralized erasure codes is illustrated in Fig. 2. The storage system has n storage servers. A data object is represented as k elements, $M_1, M_2, \ldots, M_k$ in a finite field $\mathsf{GF(p)}$ of prime order p. Each element is randomly distributed to n storage servers for v times with replacement. Each storage server picks a random coefficient from the finite field for each received element and linearly combines all received elements as a codeword element for this data object. Finally, the storage server stores the codeword element C_i and all chosen coefficients $a_{i,1}, a_{i,2}, \ldots, a_{i,k}$, where coefficients are set to 0 by default. All coefficients of all n storage servers form a generator matrix G and overall the encoding process is

$$[M_1, M_2, \ldots, M_k] \cdot G = [C_1, C_2, \ldots, C_n]$$

We illustrate the two ways of data retrieval in Fig. 3. The upper one is basic system model, which is a typical



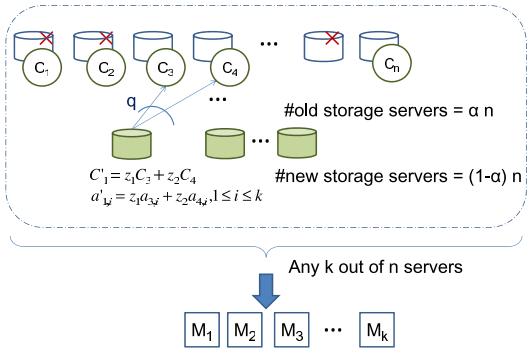


Fig. 4. The distributed repair mechanism for decentralized erasure code based distributed storage systems.

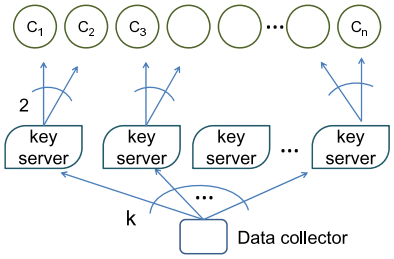


Fig. 3. Two different decoding processes of decentralized erasure codes.

method of data retrieval. A data collector queries k out of n storage servers to retrieve and decode codeword elements for the original data object. The bottom one is secure system model, which is designed for supporting data confidentiality. The storage system not only has storage servers but also key servers, who provide key management services. A data collector informs at least k key servers and each informed key server queries 2 storage servers. Then key servers forward received codeword elements to the data collector. Finally, the data collector recovers the data object back from data received from key servers.

Remark. The secure system model consisting of storage servers and key servers is firstly proposed in [4]. In that system, elements are encrypted before distributed to storage servers. In the data retrieval process, key servers perform partial decryption for the data collector. Since data confidentiality is out of our scope, we abstract the system model by ignoring the encryption and decryption processes.

In both system models, the data collector would receive codeword elements and coefficients. The condition of a successful data retrieval is that the coefficients form a $k \times k$ matrix K and the matrix K is invertible in $\mathsf{GF(p)}$. When the parameter v in the encoding process is small with respect to n, the generator matrix G is sparse, i.e., containing many 0 entries, and then the probability of a $k \times k$ sub-matrix K of G being invertible is smaller. When v is large, the communication cost on element distribution is high.

In previous studies, a family of settings on (n, k, v) is provided such that the probability of a successful data retrieval is lower bounded in both system models, as shown in Lemma 1 and Lemma 2.

Lemma 1: (Theoretical bound in basic system model) Consider a distributed storage system consisting of n storage servers in basic system model. When $n = ak^c$ with $a > \sqrt{2}$, $c \geq 1$, $v = bk^{c-1} \ln k$ and $b > 5a$, the probability of a successful data retrieval is at least $1 - k/p - \epsilon(k)$, where $\epsilon(k) = o(1)$ is

$$\left(\frac{a}{\sqrt{2}}\right)^{-k} + k^{1-b/a} + \max_{i=2}^{(k+1)/2} \{\exp\{2i(1-\ln i) + \ln k(\frac{-bi(k-i+1)}{ak} + 2i + 1)\}\} + \max_{i=2}^{(k+1)/2} \{\exp\{2i(1-\ln i) + \ln k(bik^{c-1} \ln \frac{i-1}{n} + 2i + 1)\}\} \quad (1)$$

Lemma 2: (Theoretical bound in secure system model) Consider a distributed storage system consisting of n storage servers and at least k key servers in secure system model. When $n = ak^c$ with $a > \sqrt{2}$, $c \geq 1.5$, $v = bk^{c-1} \ln k$ and $b > 5a$, the probability of a successful data retrieval is at least $1 - k/p - \epsilon(k)$.

The bounds on the probability of a successful data retrieval for the two system models are the same except that the requirement on the parameter c. In secure system model, each key server independently queries storage servers. Hence, the number of storage servers needs to be larger than one in basic system model, such that the number of storage servers queried by key servers is sufficient, i.e., at least k. As a result, it can be seen that in Lemma 2, c is set to at least 1.5 while it is set to at least 1 in Lamma 1, where $n = ak^c$.

Later, in our simulations, we enlarge the range of c for secure system model by applying the condition of $c \geq 1$. This adjustment makes the comparison of data retrievability more comprehensive between analytical values and simulation results in two system models.

B. Distributed Repair Mechanism

A distributed repair mechanism, as shown in Fig. 4 is proposed for both system models. Consider a storage system which has n storage servers initially. When $(1-\alpha)n$ storage servers fail, $(1-\alpha)n$ new storage servers join in to repair the storage system. The assumption of the repair

mechanism on the data retrievability is that a k-subset of old storage servers can recover the data object back with probability at least $1 - k/p - \epsilon(k)$. The distributed repair mechanism defines that each new storage server queries q old storage servers and picks a random coefficient from the finite field for each received codeword element and corresponding coefficients. A new storage server then linearly combines all received data and finally stores the regenerated codeword element and its coefficients. Fig. 4 shows a small example, where a new storage server queries 2 old storage servers and linearly combines two received data $(C_3, a_{3,1}, a_{3,2}, \ldots, a_{3,k})$ and $(C_4, a_{4,1}, a_{4,2}, \ldots, a_{4,k})$ with two coefficients z_1 and z_2 as $(C'_1, a'_{1,1}, a'_{1,2}, \ldots, a'_{1,k})$. Recall that a successful data retrieval is that a k-subset of n storage servers can recover the data object back.

The number q of old storage servers a new storage server has to query is critical to data retrievability. There is a tradeoff between the communication cost in terms of required connections per new storage server and data retrievability. The distributed repair mechanism is proposed with theoretical bounds on data retrievability under a family of system settings as shown in Lemma 3.

Lemma 3: (Theoretical bound for a repaired state) Let the number αn of old storage servers be k^d, where $d > 1$ is a constant. Let the number n of all storage servers be $n = ak^c$, where $a > \sqrt{2}$ and $c \geq 1$. When q is set to satisfy Equation (2), the probability of a successful data retrieval is at least $1 - 2k/p - e^{-k} - \epsilon(k)$.

$$q \geq \min\{k, \max\{\frac{2k}{(d-1)\ln k}, \frac{k}{(d-1)\ln k} + \frac{d}{d-1}\}\} \quad (2)$$

IV. Simulation Settings and Results

We derive analytical values of data retrievability from theoretical bounds and run Monte-Carlo simulations to evaluate the retrieval probability for an initial system, a system with failed storage servers, and a repaired system, respectively. Our simulations are done by using Matlab $R2011a$. We classify our simulations in two sets, Exp^{init} and Exp^{run}, which are designed for an initial state of a storage system and the other two states, respectively.

A. Simulations of Exp^{init}

Simulations of Exp^{init} are designed to evaluate data retrievability of a storage system in an initial state. There are three subsets of simulations in Exp^{init}. The first subset is to derive data retrievability from the theoretical bound in Lemma 1 and Lemma 2. The other two is to evaluate the probability in basic system model and secure system model by simulations, respectively.

We consider a set of system parameters on (a, b, c, k, p) and compute other system parameters n and v accordingly. We choose the smallest integer larger than $\sqrt{2}$ as a, i.e., $a = 2$. We then choose the smallest integer satisfying $b > 5a$ for b, i.e., $b = 11$. The robustness parameter k is considered as $8 \sim 32$. We consider c as $1, 1.5, 2, 2.5$ and 3 for various system scales n with respect to k, where $n = ak^c$. A larger p implies a better data retrievability while it also introduces computation complexity of matrix inversion.

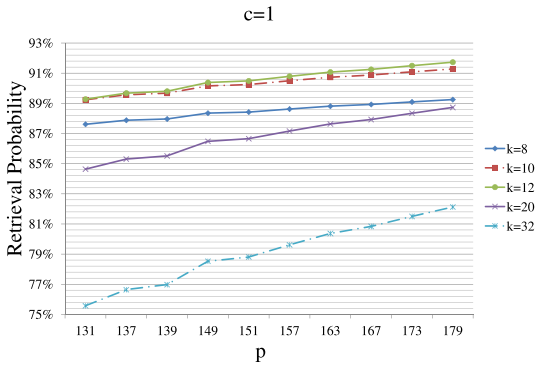


Fig. 5. Analytical values of data retrievability to observe the impact of p.

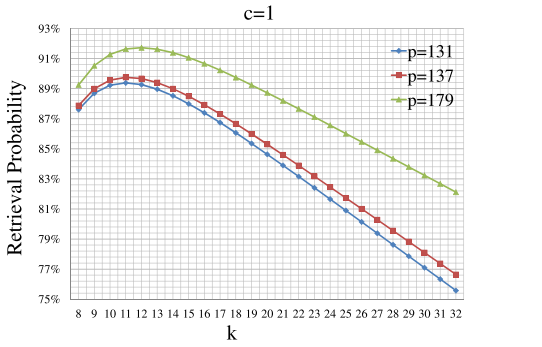


Fig. 6. Analytical values of data retrievability to observe the impact of k.

For the parameter p, we consider 10 different 8-bit primes $131, 137, 139, 149, 151, 157, 163, 167, 173$ and 179 to observe the impact of p while the computing time is acceptable in our machine.

Simulations. We simulate the random processes on coefficients in data storing and two different data retrieval in basic and secure system models, respectively. For data storing, we simulate that each element of a data object is randomly distributed among n storage servers and each server chooses a random coefficient for each received element.

For data retrieval in basic system model, we simulate that a data collector randomly chooses k storage servers to recover the data object. We simulate the choosing process iteratively. In each iteration, the data collector chooses a storage server from all n storage servers and each storage server is chosen with probability $1/n$. If the chosen storage server is chosen in previous iterations, the data collector re-chooses one until a unchosen storage server is chosen. After k iterations, the data collector chooses k distinct storage servers.

For data retrieval in secure system model, we simulate that each of k key servers chooses 2 distinct storage servers and a data collector uses the first k received data to recover the data object. Again, in each iteration, a key server chooses a storage server with probability $1/n$. A key server chooses 2 distinct storage servers by 2 iterations. The data collector then uses the first k received data of k distinct storage servers.

The received data form a square-matrix K. A successful data retrieval is the event that $(\det(K) \mod p) \neq 0$. The probability is evaluated by running the simulation for $10,000$

TABLE I. AVERAGE DATA RETRIEVABILITY OVER p FOR EVERY PAIR OF (k, c) IN BASIC SYSTEM MODEL

$k \backslash c$	1	1.5	2	2.5	3
8	99.34%	99.33%	99.35%	99.34%	99.38%
9	99.40%	99.38%	99.41%	99.40%	99.47%
10	99.96%	99.95%	99.98%	99.96%	99.96%
11∼32	100%				

times.

To speed up the simulation, we modify the random process in two places.

- For cases of $k \leq 14$, we make the modification. For each element of a data object, a storage server receives the element with probability $1-(1-1/n)^v$. In the original random process, each element is sent to a randomly chosen storage server for v times with replacement. By this modification, the number of random number generating is reduced from kv to kn, where $v > n$ for $k \leq 14$.
- When a storage server receives an element, the coefficient is firstly marked as 1. During the data retrieval, a coefficient of a queried storage server is then randomly chosen from $\mathcal{Z}_p$ if it is originally marked as 1. By this modification, the number of random numbers generating is reduced from kv to at most k^2.

B. Results of Exp^{init}

Analytical values. Under considered parameters, we derive data retrievability from the theoretical bound in Lemma 1 and Lemma 2. To observe the impact of p, we fix $c = 1$ and summarize results of 10 different p in Fig. 5 for $k = 8, 10, 12, 20$ and 32. With the same k, data retrievability is better when a large p is used. To observe the impact of k, we fix $c = 1$ and summarize results of $k = 8, 10, 20$ and 32 in Fig. 6 for $p = 131, 137$ and 179. The results show a curve that with $p = 131$, when k is set to about 11, the resulting data retrievability is the best among all k. Moreover, for a larger p, the peak of the curve occurs in a larger k. For instance, in Fig. 6, the peak for $p = 179$ occurs in $k = 12$. This observation indicates that when a system manager chooses a larger value for k, a larger finite field gives a better data retrievability. In addition, we find that the parameter c does not have impact on data retrievability since the probability of a successful data retrieval is the same with fixed k and p for all considered c.

Simulation results in basic system model are almost 100%[1] of data retrievability when k is larger than 10. With fixed c and k, we take average on the probability over all 10 considered p and summarize results in Table I. All other cases of k between 11 and 32 have probability of a successful data retrieval 100%.

We fix $c = 1.5$ and summarize results of $k = 8, 9, 10$ and 11 in Fig. 7 to observe the impact of p. We fix $k = 8$ and summarize results of $p = 131, 137$ and 179 in Fig. 8 to observe the impact of c. No significant impact of c and p is observed.

Simulation results in secure system model are almost 100% of data retrievability. With fixed c and k, we take average on the probability over all 10 considered p and summarize results in Table II. All cases of k between 11 and 32 and c ranging from 1.5 to 3 have probability of a successful data retrieval as 100%. The exception cases happened on two settings: (1) $c > 1$ and $k <= 10$ (as shown in Table II), and (2) $c = 1$ (as shown in Fig. 9). But even in the exception case, the worst data retrievability is still more than 98.4%.

[1] only few cases are 99.99%

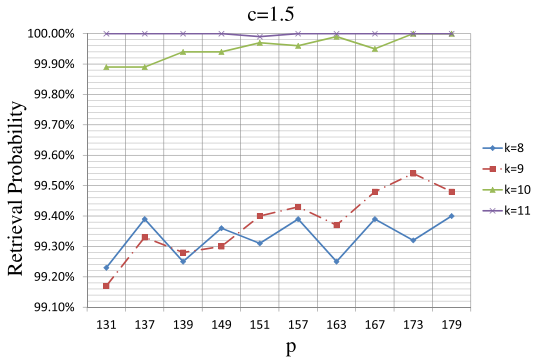


Fig. 7. Data retrievability of cases of $c = 1.5$ and $k = 8, 9$ and 10 in basic system model.

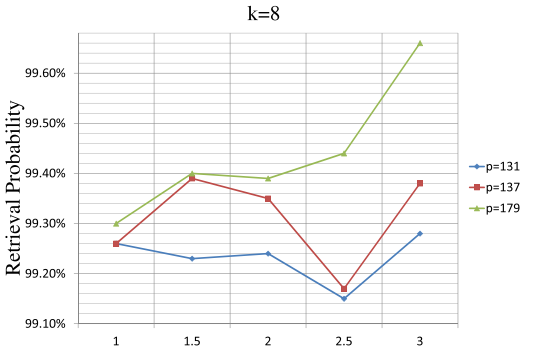


Fig. 8. Data retrievability of cases of $k = 8$ and $p = 131, 137$ and 179 in basic system model.

Comparison. To show the difference between analytical values and two sets of simulation results with the impact of k, we summarize results with fixed $p = 131$ and $c = 1$ in Fig. 10. Results show that analytical values are much smaller than simulation results, no matter in basic or secure system model. It means that the theoretical bound is not tight and implies that code based storage systems can achieve a much higher data retrievability than expected. The minimal difference over k between analytical values and simulation results is about 10%. The closest analytical value occurs when $k = 11$. The difference gets larger when k is larger.

To show the difference with the impact of p, we summarize results with fixed $k = 9$ and $c = 1$, where the difference between analytical values and simulation results is minimal, in Fig. 11. By the results, the difference gets smaller when p gets larger. It implies that the theoretical bound is more tight for a storage system using a larger finite field. Among our simulations, the minimal difference is 8.48%, which occurs when $p = 179$.

To show the difference with the impact of c, we summarize results with fixed $p = 179$ and $k = 9$ in Fig. 12. There is no

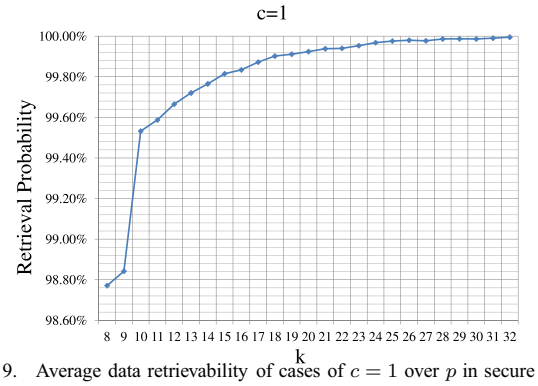


Fig. 9. Average data retrievability of cases of $c = 1$ over p in secure system model.

TABLE II. AVERAGE DATA RETRIEVABILITY OVER p FOR EVERY PAIR OF (k, c) IN SECURE SYSTEM MODEL

$k \backslash c$	1.5	2	2.5	3
8	99.35%	99.38%	99.37%	99.34%
9	99.43%	99.55%	99.44%	99.46%
10	99.95%	99.96%	99.97%	99.95%
11∼32	100%			

significant difference over c.

C. Simulations of Exp^{run}

Simulations of Exp^{run} are designed to evaluate data retrievability of a storage system with failed storage servers and a repaired storage system. We would like to know that when some storage servers fail, whether data retrievability remains $1 - k/p - \epsilon(k)$, which is the basic assumption when we apply the the distributed repair mechanism. We also investigate data retrievability of a repaired storage system while the theoretical bound gives a lower bound $1 - 2k/p - e^{-k} - \epsilon(k)$. Again, we use three subsets of simulations, analytical values and simulation results for basic and secure system models, for both system states to study the gap between theoretical bounds and simulation results.

We perform the simulation as shown in Fig. 13. We consider 30 continuous rounds which simulate 30 continuous time units in real applications. In the first round, a storage system is initialized. We consider that a storage server fails in a round with a probability f. For the state with failure, we evaluate data retrievability over surviving storage servers. Thus, data retrievability may remain high but the number of surviving storage servers in the system is decreased. The state of the system is cloned and we apply the distributed repair mechanism on the cloned state. We then evaluate data retrievability of the repaired system. The second round starts from the system containing failed storage servers. The storage system accumulates failed storage servers over rounds.

We consider a set of system parameters on (a, b, c, k, p, f) and compute other system parameters n and v accordingly. We choose the same setting for $a = 2$ and $b = 11$. We set the parameter $p = 131$ while the computing time is acceptable in our machine. The robustness parameter k is considered as $8, 10, 12, 14$ and 16. We consider c as $1, 1.5$ and 2 for various system scales n with respect to k, where $n = ak^c$. We set $f = 0.01, 0.03$ and 0.05 to simulate different corruption degrees.

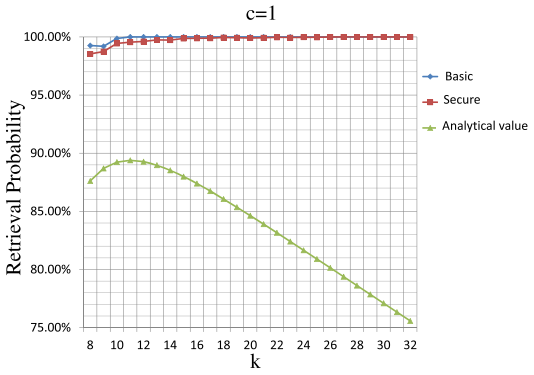


Fig. 10. Data retrievability of analytical values and simulation results with $p = 131$ and $c = 1$.

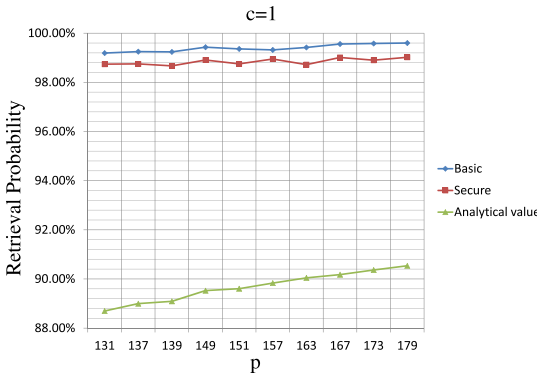


Fig. 11. Data retrievability of analytical values and simulation results with $k = 9$ and $c = 1$.

D. Results of Exp^{run}

We firstly probe data retrievability of a storage system with failed storage servers. We are interested in when a storage system satisfies the assumption of the distributed repair mechanism, which requires data retrievability of the system to be $1 - k/p - \epsilon(k)$. We fixed $c = 1$ and $k = 8$ and summarize analytical values and simulation results in basic and secure system models in Fig. 14. Results are summarized by the fraction of surviving storage servers. We identify threshold values which indicate when a storage system satisfies the required data retrievability. When the fraction of surviving storage servers is over the threshold, the distributed repair mechanism is applicable. Additionally, when the fraction is getting close to the threshold from 100%, the storage system needs to be repaired. In Fig. 14, the threshold for basic system model is 62% and for secure system model is 74%. We observe that in secure system model, the fraction of surviving storage servers needs to be larger than one in basic system model to meet the requirement of applying the distributed repair mechanism.

We summarize threshold values of all considered system parameters in basic and secure system models in Table III, where "always" means that simulation results always satisfy analytical values. From Table III, we make the following three observations.

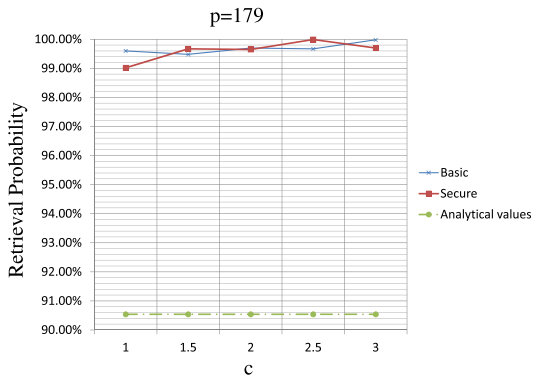


Fig. 12. Data retrievability of analytical values and simulation results with $p = 179$ and $k = 9$.

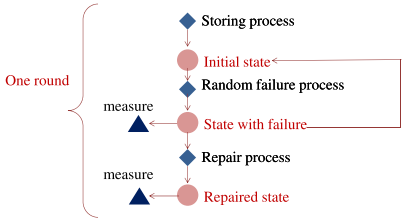


Fig. 13. Simulation processes of a storage system from an initial state through a state with failure to a repaired state.

- With a fixed k, the threshold value is smaller as c is larger. When $c = 1.5$ or $c = 2$, the threshold value is smaller as k is larger. It implies that a storage system with less storage servers need to be repaired more often.
- When $c = 1$, threshold values are almost the same over k. It indicates that when a system manager sets $c = 1$, he can consider the same condition on the fraction of surviving storage servers for executing the distributed repair mechanism.
- Threshold values in secure system model are at least ones in basic system model. It implies that with fixed c and k, a storage system in secure system model needs to be repaired more often.

We also probe data retrievability of a repaired storage system. We are interested in knowing under which conditions a storage system can be repaired such that the repaired storage system has data retrievability as in Lemma 3. Similarly, we fixed $c = 1$ and $k = 8$ and summarize analytical values and simulation results in basic and secure system models in Fig. 15 according to the fraction of surviving storage servers. We identify threshold values on the fraction of surviving storage servers, where they indicate the conditions we are looking for. A larger threshold implies that a storage system has to execute the distributed repair mechanism earlier than when the fraction of surviving storage servers hits the threshold. Hence the storage system needs to be repaired more often to achieve data retrievability satisfying the theoretical bound. In Fig. 15, the thresholds for basic and secure system model are both 60%.

Similarly, we summarize threshold values of all considered

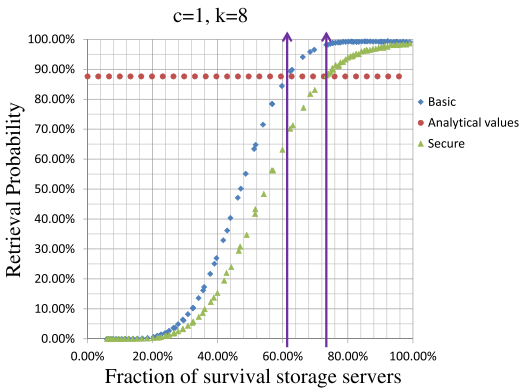


Fig. 14. Comparison on data retrievability of a state with failure by the theoretical bound and simulation results.

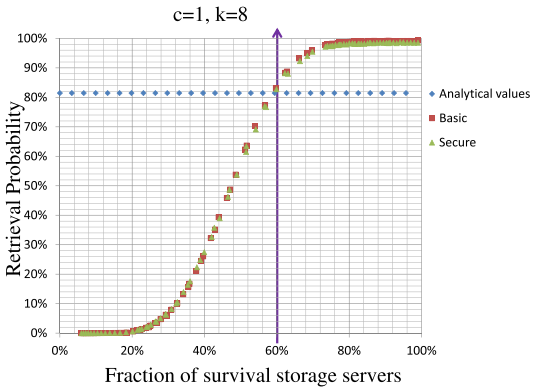


Fig. 15. Comparison on data retrievability of a repaired state by the theoretical bound and simulation results.

system parameters in basic and secure system models in Table IV. We then make the following obervations.

- With a fixed k, when c gets larger, threshold value is smaller. With a fixed c, threshold values are smaller when k gets larger. It means that when k is fixed and the number n of storage servers is larger, the requirement on the fraction of surviving storage servers to achieve theoretical data retrievability is decreased.
- With $c = 1$, threshold values are close to each other. The maximal difference is 4%.

By comparing Table III and Table IV, we can see whether the assumption or the theoretical bound is tight. The threshold 62% of $c = 1$ and $k = 8$ in basic system model in Table III means that when the fraction of surviving storage servers is at least 62%, the storage system satisfies the assumption. By Lemma 3, after the system is repaired, the resulting data retrievability is at least $1 - 2k/p - e^{-k} - \epsilon(k)$. Meanwhile, the threshold 60% of $c = 1$ and $k = 8$ in basic system model in Table IV means that when the fraction of surviving storage servers is at least 60%, the storage system can be repaired such that the resulting data retrievability is at least $1 - 2k/p - e^{-k} - \epsilon(k)$. As a result, we can observe that the

TABLE III. SUMMARY OF THRESHOLD VALUES IN BASIC AND SECURE SYSTEM MODELS FOR A STATE WITH FAILURE

threshold values in basic system model					
$c \backslash k$	8	10	12	14	16
1	62%	63%	62%	60%	60%
1.5	24%	22%	14%	18%	17%
2	10%	7%	7%	always	always
threshold values in secure system model					
$c \backslash k$	8	10	12	14	16
1	74%	76%	75%	74%	74%
1.5	30%	26%	20%	22%	20%
2	12%	9%	7%	always	always

TABLE IV. SUMMARY OF THRESHOLD VALUES IN BASIC AND SECURE SYSTEM MODELS FOR A REPAIRED STATE

threshold values in basic system model					
$c \backslash k$	8	10	12	14	16
1	60%	60%	57%	57%	56%
1.5	22%	20%	18%	17%	16%
2	9%	7%	always	always	always
threshold values in secure system model					
$c \backslash k$	8	10	12	14	16
1	60%	60%	57%	57%	56%
1.5	26%	24%	18%	17%	18%
2	9%	8%	always	always	always

assumption sets a more restricted condition on data retrievability. In other words, the theoretical bound in Lemma 3 is not tight.

We observe that for a repaired system, data retrievability derived from the theoretical bound is less than 90%. We are interested in knowing conditions of a storage system such that the repaired system has data retrievability more than 95%. We re-plot lines to find threshold values in basic and secure system models. Fig. 16 shows results of $c = 1$ and $k = 8$, where threshold values for basic and secure system models are both 70%. We summarize results of thresholds in Table V. It provides a reference for a system manager to decide under what condition to execute the distributed repair mechanism to have the repaired system with data retrievability 95%. For instance, consider that a storage system with $k = 8$ has n storage servers, where $n = 16$ (i.e. $a = 2$ and $c = 1$). A system manager can monitor the fraction of surviving storage servers in the storage system and decide to execute the repair mechanism when the fraction is approaching 70% from 100%. As a result, the manager maintains the system with data retrievability at least 95%.

V. SUMMARY AND CONCLUSION

We summarize our empirical study to address data retrievability of decentralized erasure code based storage systems as follows.

- For a storage system in an initial state, data retrievability is more than 99% except for the setting $c = 1$ in secure system model. Even in the exception case, the worst data retrievability is more than 98.7%. We conclude that the gap between simulation results and theoretical bound is significant. The minimal difference among our results is 8.48%.
- For the distributed repair mechanism, we characterize when a storage system can keep satisfying the assumption on data retrievability. Among our results, 76% is the highest threshold for the fraction of surviving storage servers for satisfying the assumption. The results provide a reference for a system manager to decide when to execute the repair mechanism.
- For the distributed repair mechanism, we characterize when a storage system can be repaired such that the resulting data retrievability is over the theoretical bound. The bound is 80% in our cases. Among our results, the highest threshold is when the fraction of surviving storage servers is 60%. We also characterize when a storage system can be repaired such that the resulting data retrievability is at least 95%. When a system with at most 70% storage servers being survival, it can be repaired by the distributed repair mechanism such that the resulting data retrievability is at least 95%.
- We compare characteristics of when a storage system can keep satisfying the assumption and when a storage system can be repaired such that the resulting data retrievability is over the theoretical bound. By the comparison, we observe that the assumption results in a slightly higher threshold value. Moreover, the difference in secure system model is larger than one in basic system model. Among all our results, the biggest difference is 5% in basic system model and 16% in secure system model, respectively. We conclude that the assumption for basic system model is tighter than for secure system model.

By our empirical study on data retrievability in decentralized code based distributed storage systems, we conclude

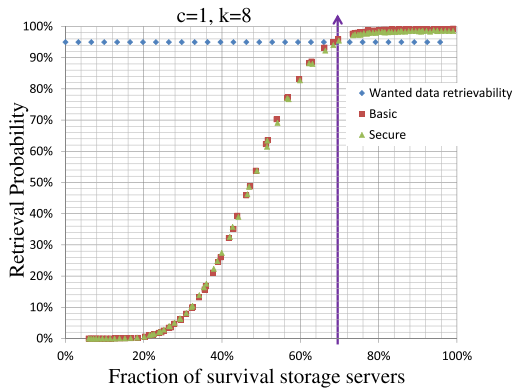


Fig. 16. Threshold values on fraction of surviving storage servers for achieving data retrievability 95%.

TABLE V. SUMMARY OF THRESHOLD VALUES FOR A REPAIRED STATE WITH DATA RETRIEVABILITY MORE THAN 95%

threshold values in basic system model					
$c \backslash k$	8	10	12	14	16
1	70%	67%	66%	66%	64%
1.5	27%	24%	22%	20%	18%
2	11%	8%	7%	always	always
threshold values in secure system model					
$c \backslash k$	8	10	12	14	16
1	70%	67%	66%	66%	64%
1.5	34%	30%	22%	20%	22%
2	11%	10%	6%	always	always

that data retrievability is underestimated by theoretical bounds for an initial system and for a repaired system. For an initial system, the gap between analytical values derived from theoretical bounds and simulation results is significant. By our simulation results, data retrievability is over 98% for an initial system while analytical values are less than 90%. For a repaired system, the gap is less significant. Data retrievability is over 80% for a repaired system while analytical values are about 80%. Our simulations are operated in a finite field of a 8-bit prime order. Since a larger finite field gives a higher data retrievability, a storage system using a larger finite field would have a better data retrievability than our simulation results. We conclude that decentralized erasure code based storage systems are practical in terms of data retrievability. Moreover, our simulation results provide some references for a system manager to decide system parameters and under what conditions to execute the distributed repair mechanism.

ACKNOWLEDGMENT

The research was supported in part by projects ICTL-102Q707, ATU-102W958, NSC 101-2218-E-009-003 and NSC 101-2218-E-009-008.

REFERENCES

[1] A. G. Dimakis, 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] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[3] D. Leong, A. G. Dimakis, and T. Ho, "Distributed storage allocation for high reliability," in *Proceedings of IEEE International Conference on Communications - ICC'10*. IEEE, 2010, pp. 1–6.

[4] H.-Y. Lin and W.-G. Tzeng, "A secure decentralized erasure code for distributed network storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, pp. 1586–1594, 2010.

[5] ——, "A secure erasure code based cloud storage system with secure data forwarding," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 6, pp. 995–1003, 2012.

[6] H.-Y. Lin, W.-G. Tzeng, and B.-S. Lin, "A decentralized repair mechanism for decentralized erasure code based storage systems," in *Proceedings of the IEEE International Conference on Trust, Security and Privacy in Computing and Communications - TrustCom'11*, 2011, pp. 613–620.

[7] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE/ACM Transactions on Networking*, vol. 14, pp. 2809–2816, 2006.

[8] Y. Wu, A. G. Dimakis, and K. Ramchandran, "Deterministic regenerating codes for distributed storage systems," in *Proceedings of the 45th Annual Allerton Conference on Communication, Control, and Computing - Allerton'07*, 2007, pp. 1243–1249.

[9] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing - Allerton'09*, 2009, pp. 1243–1249.

[10] N. B. Shah, K. V. Rashmi, and P. V. Kumar, "A flexibile class of regenerating codes for distributed storage," in *Proceedings of IEEE Symposium on Information Theory - ISIT'10*, 2010, pp. 1943–1947.

[11] S. Akhlaghi, A. Kiani, and M. R. Ghanavati, "A fundamental trade-off between the download cost and repair bandwidth in distributed storage systems," in *Proceedings of IEEE International Symposium on Network Coding - NetCod'10*, 2010, pp. 1–6.

[12] S. E. Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *Proceedings of the 48th Annual Allerton Conference on Communication, Control, and Computing - Allerton'10*, 2010, pp. 1510–1517.

[13] Q. Yu, K. W. Shum, and C. W. Sung, "Minimization of storage cost in distributed storage systems with repair consideration," in *Proceedings of the Global Communications Conference - GLOBECOM'11*, 2011, pp. 1–5.

[14] X. Wang, Y. Xu, Y. Hu, and K. Ou, "Mfr: Multi-loss flexible recovery in distributed storage systems," in *Proceedings of the IEEE International Conference on Communications - ICC'10*, 2010, pp. 1–5.

[15] Y. Hu, Y. Xu, X. Wang, C. Zhan, and P. Li, "Cooperative recovery of distributed storage systems from multiple losses with network coding," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 268–276, 2010.

[16] K. W. Shum, "Cooperative regenerating codes for distributed storage systems," in *Proceedings of the IEEE International Conference on Communications - ICC'11*, 2011, pp. 1 –5.

[17] F. E. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," in *Proceedings of the 30th IEEE International Conference on Computer Communications - INFOCOM'11*, 2011, pp. 1215–1223.

[18] A.-M. Kermarrec, N. L. Scouarnecy, and G. Straub, "Repairing multiple failures with coordinated and adaptive regenerating codes," in *Proceedings of International Symposium on Network Coding - NetCod'11*, 2011, pp. 1–6.

[19] K. W. Shum and Y. Hu, "Existence of minimum-repair-bandwidth cooperative regenerating codes," in *Proceedings of International Symposium on Network Coding - NetCod'11*, 2011.

[20] ——, "Exact minimum-repair-bandwidth cooperative regenerating codes for distributed storage systems," in *Proceedings of IEEE International Symposium on Information Theory - ISIT'11*, 2011, pp. 1442–1446.