Fault-Tolerant and Secure Distributed Data Storage Using Random Linear Network Coding

Pouya Ostovari and Jie Wu
Department of Computer & Information Sciences
Temple University, Philadelphia, PA 19122
Email: {ostovari, jiewu}@temple.edu

***Abstract*—Network coding is a technique which can be used in wired and wireless networks to increase the throughput of the networks and provide reliable transmissions. Also, it can be used in distributed storage systems to store a large data on different storages and provide fault tolerance against storage failures. Using network coding, the set of packets that form a file can be encoded to infinite number of packets, and a subset of these coded packets is sufficient to retrieve the original data. In addition to provide fault tolerance, network coding is an efficient tool to protect the data from eavesdroppers. An eavesdropper is not able to decode the coded packets and retrieve the original data unless it has access to a sufficient number of coded packets. Increasing the redundancy enhances the fault tolerance. However, it makes the system more vulnerable against eavesdropper attacks. In this work, we perform a trade-off between security of a distributed storage system and its fault tolerance. We formulate the problem as a mixed integer and linear programming, and propose two linear programming optimizations to solve it.**

***Index Terms*—Network coding, security, fault tolerance, data storage, optimization, random linear network coding.**

I. Introduction

These days we are witnessing a fast increase in the popularity of the cloud computing and distributed storage systems. These distributed storages are more convenient than the local backup storages. Moreover, they are more reliable and secure than the personal hard drives. The distributed storage systems, such as Dropbox, GoogleDrive, and OneDrive, store the files on multiple storages, which provides fault tolerance. In the case that a few storages fail, the redundant stored files on the other storages can be used to retrieve the original data. On the other hand, these distributed storage systems store different versions of the files in the case that they are modified.

One of the main challenges in distributed storage systems is providing fault tolerance. In order to provide fault tolerance, redundant data needs to be stored on multiple storages. It is clear that adding more redundancy provides a higher level of protection against storage failure. However, more redundancy requires more storage, which increase the cost. There are many existing work on fault-tolerant distributed storages that addressed the problem finding the amount of required redundancy to achieve a given level of fault tolerance. One of the techniques that can be used to construct redundancy and as a result fault tolerance is random linear network coding. In random linear network coding, the original packets are coded with each other linearly and using random coefficients. Assuming that we have m packets, we can potentially generate infinite number of coded packets, and any m linearly independent coded packets are sufficient to decode and retrieve the original packets. The decoding is performed by solving a system of linear equations using Gaussian elimination

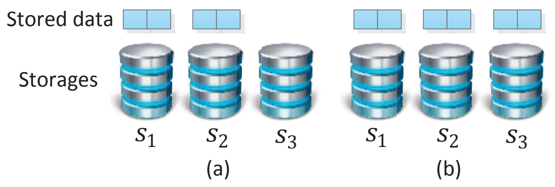


Fig. 1. Motivation example.

Consider a scenario in which a large file needs to be stored securely over a set of distributed networks. The networks are not trustworthy, which means that an eavesdropper might access to some of them. An straightforward way to prevent the eavesdropper to reconstruct the original data from the subset of the storages that are accessed is to use cryptographic methods. For this purpose, the original data can be encrypted using a secret key. Then, the encrypted data can be partitioned to multiple packets and stored on the untrusted data storages. However, the problem of the cryptographic methods is their complexity.

For this reason, the work in [1], [2] propose a method that relies on network coding rather than cryptography. The main idea of the paper is that, random linear network coding can provide fault tolerance and security at the same time. In the case that the eavesdropper does not have access to a sufficient number of coded packets, it cannot use Gaussian elimination to decode the coded packets and construct the original data. As a result, we only need to make sure that the eavesdropper does not get enough coded packets. In this way, we can achieve confidentiality without additional bandwidth or storage space costs. In these work, only the authorized users know the location of the data storages. Therefore, the data storage locations plays the role of secret key that is used in the cryptographic methods. The authors in [1] show that this technique guarantee robustness against failures and eavesdroppers at the same time.

In [1], [2], it is assumed that an eavesdropper have access to at most a given number of data storages. Consequently, it is easy to find a content distribution method that makes sure that an eavesdropper cannot access to k coded packets. However, if the eavesdropper get access to more number of data storages, it will be able to retrieve the original data. In this work, we consider a different model. We assume that we do not know the number of data storages that can be accessed by an eavesdropper. However, we know the vulnerability of each data storage against eavesdropper attacks. It is clear that adding more redundancy and storing more coded packets increases the fault tolerance of the system. However, it also increases the vulnerability of the system against eavesdroppers.

Consider the example shown in Figure 1. Assume that the failure probability of storages s_1, s_2, and s_3 equal 0.1. The eavesdropping probability of storages s_1 and s_2 are equal to 0.1. Moreover, the eavesdropping probability of storage s_3 is 0.2. A file which contains 4 packets is stored on the 3 data storages. In Figure 1(a), we store 2 packets on each of data storages s_1 and s_2. In this case, the file can only be retrieved by a user when none of the storages s_1 and s_2 fails. If we define the reliability of the system as the probability of successful data retrieval, that equals to $0.9^2 = 0.81$. Moreover, an eavesdropper needs to access both of s_1 and s_2 to retrieve the data. As a result, the probability of eavesdropping is equal to $0.1^2 = 0.01$. Now consider the example in Figure 1(b), in which 2 packets are stored on each of the three storages. In this case, accessing to any 2 out of the 3 data storages is sufficient to retrieve the data by a user of an eavesdropper. Consequently, the reliability of the system equals $0.9^3 + 3 \times 0.9^2 \times 0.1 = 0.972$. Also, the eavesdropping probability equals $0.1^3 + 2 \times 0.1^2 \times 0.9 = 0.019$. By comparing this two cases, we find that adding more redundancy increases the reliability of the data storage system. However, it also increases the vulnerability of the system against eavesdropping attacks.

This example shows that, as we increase the redundancy, the storage system becomes more robust against storage failures. On the other hand, more redundancy increase the vulnerability of the storage system against eavesdropping attacks. As a result, when we want to address both of the system reliability and the security without using a cryptographic method, we need to perform a trade-off between these two objectives. This trade-off should consider the amount of the data redundancy and the amount of stored data on each data storage. A data storage might be reliable in terms of failure; however, that might not be secure.

Motivated by the example in Figure 1, we study the problem of fault-tolerant and secure distributed data storage in this paper. In order to store redundant data, we apply random linear network coding on the original data, and store the coded packets on the storages. In order to address reliability and security, we perform a trade-off between them. We formulate the problem as a mixed integer and linear programming. It is know that mixed integer and linear programming optimizations are NP-hard in general. In order to reduce the complexity of the optimization, we relax the mixed integer and linear programming optimization to linear programming, which can be solved in polynomial time. We evaluate our proposed methods using simulations.

The remainder of this paper is organized as follows. In Section II, we review the related work and provide a background on network coding and its applications. We discuss the system model, assumptions, and our objective in Section III. In Section IV, we present our secure and reliable data storage method, which is a trade-off between security and reliability. We present the simulation results in Section V. Section VI concludes the paper.

II. Related Work and Background

In this section, we review the related work and discuss the preliminaries. Network coding is the core part of this work. For this reason, we first discuss the related work on network coding. Then, we provide a background on network coding. We also discuss the some of the applications of network coding, including security and reliability.

A. Network Coding Preliminaries

The authors in [3] proposed the idea of network coding [4]–[6] for the first time. They show that network coding helps to achieve the capacity of a single multicast session in wired networks, which is equal to the min-cut max-flow between the source node and the destination nodes. This can be achieved by solving the bottleneck problem in wired networks. In [7], it is shown that to achieve the capacity of a single multicast problem, we can use linear network coding, in which the coded packets are linear combination of the original packets. The authors in [8] proposed random linear network coding for the first time. The main idea in random linear network coding is to use random coefficients to code the packets. The main advantage of the random linear network coding is that each intermediate node can perform coding independent of the other nodes. As a result, the coding can be perform in a distributed fashion. Also, it simplifies the coding process. The authors in [8] show that a high probability the generated random linear coded packets are linearly independent. An algebraic model of the linear network coding is proposed in [9].

The coefficients in random linear network coding are chosen randomly over a finite field (Galois field). Also, the linear operations are performed over a finite field. Each packet is in the form of $\sum_{j=1}^{m} \alpha_j \times P_j$. In this equation, P_j and α_j are the packets that are coded with each other and the random coefficients, respectively. The packets that are coded with each other can be plain packets (original packets) or coded packets. An important property of linear network coding is that even if coded packets are recoded with each other, the result will be a linear coded packet. As a result, each intermediate node can perform coding (recoding) on the received packets and forward the coded packets. Each destination node will be able to decode the received linearly coded packets once it receives m linearly independent coded packers. For decoding the codded packets, Gaussian can be used to solve a system of linear equations.

B. *Applications of Network coding*

1) Reliable Transmission: One of the main applications of network coding is in providing reliable transmissions in wired and wireless networks. The simplest way to provide reliable transmission is to use feedback messages. For this purpose, ARQ (automatic repeat request) method [10] can be used. Feedback messages has overhead, since for each set of packets, the source node needs to stop transmitting packets and listen to the feedback messages. This overhead becomes a major problem in the case of multicast applications, which makes ARQ almost impractical. One solution to reduce the number of feedback messages is to use hybrid-ARQ methods [11], [12], in which ARQ and forward error correcting codes (FEC) [13]–[15] are combined. In FEC methods, redundancy is added to the transmissions to fight with the transmissions error and increase the success delivery rate. However, the hybrid-ARQ methods do not eliminate the need for feedback messages

The main benefit of linear network coding is that all of the coded packets contribute the same amount of information to the destination nodes. As a result, it is not important to know which packets have been received by a destination node and which of them have been lost. Instead, in order to provide reliable transmissions we need to make sure that the destination node receives a sufficient number of coded packets. The source node and the intermediate nodes keep transmitting random linear network coded packets, until the destination node decode the coded packets. Consequently, there will be no need for feedback messages.

The application of network coding in providing reliable transmissions and reducing the number of transmissions has been received a lot of attention from the community. In [16]–[20], network coding is used in one-hop multicasting application to reduce the number of transmissions. The destination nodes transmit feedback messages to inform the source node about the lost packets. The source node uses network coding in the retransmission phase to reduce the number of required transmissions to deliver the missed packets. The idea is to combine the packets that are missed by a destination node (set of nodes) and has been received by the other nodes together. As a result, each coded packet can deliver multiple lost packets to different destination nodes. In this application, network coding helps to reduce the number of required transmissions, which results in increase in the throughput of the system.

2) Protocol Simplification: Network coding can be used as a tool to simplify some protocols in wired and wireless networks. For an instance, in per-ro-peer (P2P) networks [21]–[23], large data is stored on a set of distributed devices. In order to retrieve the original data, we need to download different parts of the data from different devices. As a result, we need to have a mechanism to track which the data that is stored on each device, which is a major challenge in P2P systems. When network coding is applied on the original data, tracking the stored data becomes easier [24]. In this case, instead of knowing which part of the data has been stored on which peer, we need to know how many coded packets are stored on the peer. In addition, network coding can simplify content distribution over distributed storage systems. Many content distribution problems are NP-complete, which means they cannot be solved in polynomial time, e.g. the problems in [25], [26]. Network coding can help to convert these problems to a flow maximization problem, which can be solved in polynomial time with linear programming optimization [27], [28]. The original problems without network coding were integer programming optimizations.

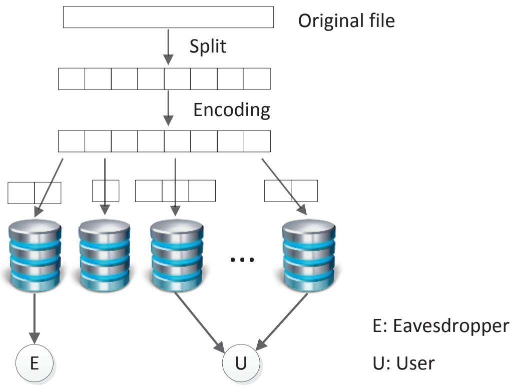


Fig. 2. System model.

3) Security: Network coding is a natural way to conceal data from unauthorized users, since a sufficient number of coded packets are required for decoding the coded packets. Therefore, it can be used to provide a lightweight secure data transmissions or data storages against eavesdroppers. If we code m original packets together, any node including an eavesdropper needs to collect m coded packets to be able to decode the coded packets and retrieve the original data. Assume that the eavesdropper has only access to $m-1$ coded packets. In this case, Gaussian elimination will fail and the eavesdropper cannot retrieve the original data. Not only network coding can hide the original data from the malicious nodes, but also can hide it from the intermediate relay nodes. If we do not apply network coding on the original data, an encryption method needs to be used to protect the data.

III. System Model

We consider a distributed data storage system, which contains n storages. Each of these data storages might fail with a given probability. The failure can be due to different reasons, such as power limitation, hardware problem, or high workload. We represent the failure probability of the ith data storage as ϵ_i. Moreover, each data storage is subject to eavesdropping attack. We represent the probability that an eavesdropper can have access to the contents of the ith data storage as γ_i. We have a large file with size m packets, and we want to store the file on the data storages. The system setup is shown in Figure 2.

TABLE I
THE SET OF SYMBOLS USED IN THIS PAPER.

Notation	**Definition**
m	Number of packets in the original file
n	Number of data storages
ϵ_i	Failure probability of the ith storage
γ_i	Access probability of the eavesdropper to the ith data storage
s_j	The jth subset of the storages
p_j	Failure probability of the storages in set s_j
q_j	Access probability of the eavesdropper to the data storages in set s_j
x_i	Portion of stored file on the ith storage
y_j	Boolean variable which shows whether set s_j stores m coded packets
U	Utility function
u_1/u_2	The security/fault tolerance of the system
α_1/α_2	The assigned weights to security/fault tolerance
t_1/t_2	Threshold for security/fault tolerance

In order to provide fault tolerance, random linear network coding is applied on the original file, and network coded packets are stored on the data storages. For this purpose, the file is divided into m packets, and random linear network coding is performed among the m packets to generate coded packets. In this way, we can store redundant coded packets. A user needs to have access to any m coded packets to be able to decode the coded packets and retrieve the original file.

Our objective is to perform a trade-off between the reliability and the robustness against the eavesdropper. A data storage that is robust against failure, might not be as secure as the other data storages. Moreover, in order to make the system more robust against failures, we need to add more redundancy and store the file on more number of data storages. However, more number of data storages make the system more vulnerable against eavesdroppers, since there is a higher chance that an eavesdropper can access m coded packets.

We represent the probability that the eavesdropper and an authorized user can retrieve the original file as u_1 and u_2, respectively. We use α_1 and α_2 as tunable weights to perform a trade-off between u_1 and u_2. In our model, variable x_i represents the portion of the file store on storage s_i. The set of subsets of the storages is denoted as R, and the jth subset in represented as R_j. Boolean variable $y_j = 1$ means that the stored data on the storages in R_j are greater than m, which is sufficient to retrieve the original file. The set of notations used in this work are shown in Table I.

IV. SECURE DATA STORAGE

As discussed in the previous section, our objective is to perform a trade-off between security and reliability. In this section, we first formulate out problem as a mixed integer and linear programming optimization. We then propose our methods to find a secure and fault tolerance method to store the file on the data storages.

A. Formulation

We formulate our problem in the following three cases.

- Case 1: We define the objective function as a function of fault tolerance and security, and optimize them at the same time.
- Case 2: We fix the fault tolerance into a specific threshold, and set it as a constraint of the optimization. This threshold is the minimum required fault tolerance that is required for a given storage system. We then minimize the eavesdropping probability such that the fault tolerance constraint is meet.
- Case 3: This is the opposite of Case 2. We define a eavesdropping probability threshold and set it as a constraint. Then, we maximize the fault tolerance, such that the eavesdropping probability constraint is meet.

1) Case 1: Our objective is to perform a trade-off between security and fault tolerance. Storing more redundant data on each data storage makes the system more robust gainst storages failure. On the other hand, it increase the vulnerability of the system against eavesdropping attacks.

We define the objective function as $U = \alpha_1 u_1 - \alpha_2 u_2$. Here, u_1 is the probability that the eavesdropper can retrieve the original file. Moreover, u_2 is the reliability of the data storage system, which is defined as the probability that a authorized user can retrieve the original file. Also, α_1 and α_2 are the tunable weights that are assigned to u_1 and u_2. The purpose of these weights are to perform a trade-off between the security and the reliability of the system.

In order to retrieve the original file, the eavesdropper need to receive m linearly independent coded packets. Then, it can use Gaussian elimination to decode the coded packets. In the case that the size of the finite field is sufficiently large, any m out of the random linear coded packets will be linearly independent and sufficient to retrieve the original file with a hight probability [8]. Assume that R is the set of subsets of the data storages. We represent the jth subset in R as R_j. We use boolean variables y_j and z_j to shows whether the eavesdropper and a user can retrieve the original file from the set of storage nodes in R_j. In the case that at least m coded packets are stored on the set of storages in R_j, y_j and z_j have a values equal to 1; otherwise, they equal 0. As a result, the probability that and eavesdropper and a user can use the set of data storages in R_j ro retrieve the file equal $q_j y_j$ and $p_j z_j$, respectively. The portion of the file that is stored on data storage s_i is equal to x_i. We can formulate our problem as the following optimization problem:

$$\min\ U = \sum_{R_j \in R} \alpha_1 q_j y_j - \alpha_2 p_j y_j \tag{1}$$

$$s.t \quad y_j \in \{0, 1\} \quad \forall\ R_j \in R \tag{2}$$

The probability that storage s_i fails is equal to ϵ_i. As a result, the probability that the storages in R_j do not fail and the rest of the storages fail can be calculated as:

$$p_j = \prod_{s_k \in R_j} (1 - \epsilon_k) \prod_{s_i \notin R_j} \epsilon_i \tag{3}$$

Moreover, the probability that an eavesdropper can have access to storage s_i equals γ_k. Therefore, the probability that an eavesdropper has only access to the set of storages in R_j can be calculated as follows:

$$q_j = \prod_{s_k \in R_j} \gamma_k \prod_{s_i \notin R_j} (1 - \gamma_i) \tag{4}$$

2) Case 2: In the second scenario, our goal is to minimize the vulnerability of the storage system against eavesdropping. We also want to achieve a certain level of fault tolerance. We show this threshold as t_2. In this case, we can formulate the problem as the following mixed integer and linear programming:

$$\min \ U = \sum_{R_j \in R} q_j y_j \tag{5}$$

$$s.t \quad \sum_{R_j \in R} p_j y_j \geq t_2 \tag{6}$$

$$y_j \in \{0, 1\} \quad \forall \ R_j \in R \tag{7}$$

here, the objective function is minimizing $\sum_{R_j \in R} q_j y_j$, which is the probability of successfully decoding the coded packets by the eavesdropper. Moreover, Constraint (6) ensures that the reliability of the data storage system is not less than threshold t_2. In Constraint (6), $\sum_{R_j \in R} p_j y_j$ is the probability that the original file can be retrieved from the storages.

3) Case 3: In the third case, our objective is to maximize the system fault tolerance against failures. However, we need to make sure that the vulnerability of the system is not more that a threshold, denoted as t_1. The mixed integer and linear programming in this case is as follows:

$$\max \ U = \sum_{R_j \in R} p_j y_j \tag{8}$$

$$s.t \quad \sum_{R_j \in R} q_j y_j \leq t_1 \tag{9}$$

$$y_j \in \{0, 1\} \quad \forall \ R_j \in R \tag{10}$$

where, the objective function is $\sum_{R_j \in R} p_j y_j$. Also, we have Constraint (6) to make sure that the probability of sucessful eavesdropping is not greater than t_1.

B. File Distribution Scheme

1) Relaxation to Linear Programming: The optimization in the previous section is and mixed integer and linear programming. In general, there is no polynomial solution for mixed integer and linear programming. In contrast, there are efficient solutions in polynomial time for linear programming optimizations. One of the techniques that can be used in order to make a mixed integer and linear optimization easier to solve, is to relax it to linear programming. For this purpose, we can relax variable y_j to a variable z_j with real values. In this case, our optimization becomes:

$$\min \ U = \sum_{R_j \in R} \alpha_1 q_j z_j - \alpha_2 p_j z_j \tag{11}$$

$$s.t \quad z_j = \sum_{i: s_i \in R_j} x_i \quad \forall \ R_j \in R \tag{12}$$

$$z_j, x_i \in (0, 1) \quad \forall \ R_j \in R, s_i \in S \tag{13}$$

We denote this optimization as LP1. This optimization can be solved using standard optimization techniques such as gradient approach. The complexity of the solutions for linear programming problems is a polynomial function of number of its variables and constraints. As a result, in the case that we have a large number of storages, the time complexity of finding its optimal solution becomes exponential. For this purpose, instead of solving the above optimization, we use its approximation as follows:

$$\min \ U = \sum_{s_i \in S} \alpha_1 \gamma_i x_i - \alpha_2 (1 - \epsilon_j) x_i$$

$$s.t \quad x_i \in (0, 1) \quad \forall s_i \in S$$

We denote this optimization as LP2.

For the second case, the linear programming optimization becomes:

$$\min \ U = \sum_{R_j \in R} q_j z_j \tag{14}$$

$$s.t \quad \sum_{R_j \in R} p_j z_j \geq t_2 \tag{15}$$

$$z_j = \sum_{i: s_i \in R_j} x_i \quad \forall \ R_j \in R \tag{16}$$

$$z_j, x_i \in (0, 1) \quad \forall \ R_j \in R, s_i \in S \tag{17}$$

In order to maximize the fault tolerance of the system and meet the security threshold, we can formulize the problem as the following linear programming optimization:

$$\min \ U = \sum_{R_j \in R} p_j z_j \tag{18}$$

$$s.t \quad \sum_{R_j \in R} q_j z_j \geq t_1 \tag{19}$$

$$z_j = \sum_{i: s_i \in R_j} x_i \quad \forall \ R_j \in R \tag{20}$$

$$z_j, x_i \in (0, 1) \quad \forall \ R_j \in R, s_i \in S \tag{21}$$

2) Greedy Algorithm: In order to further reduce the complexity of the distribution algorithm, heuristic methods such as greedy algorithms can be used. Here, we discuss the high-level idea for a greedy approach that can be used to perform a trade-off between security and reliability in distributed storage systems. We first calculate $g_i = (1 - \epsilon_i)/\gamma_i$ for each data storage s_i. This metric shows the rate of the reliability of a data storage to its vulnerability against eavesdroppers. A

Algorithm 1 Search for Optimal Coding Scheme

```
1: For each storage $s_i$ $g_i = (1 - \epsilon_i)/\gamma_i$
2: for Each storage $s_j$ in decreasing order of $g_j$ do
3:     Set $x_i = f(\epsilon_i, \gamma_i, \alpha_2, \alpha_1)$
4: end for
```

higher value for g_i means that the storage s_i is a better choice to store the coded packets, since s_i is more robust against failures and eavesdroppers. We then sort the data storages in decreasing order of their g_i, and fill them in this order.

The factors that can effect the amount of stored data of each storage s_i are ϵ_i, γ_i, α_1, and α_2. As a result, we need to have a function to these parameters to calculate x_i for each storage. For this purpose, we store $x_i = f(\epsilon_i, \gamma_i, \alpha_2, \alpha_1)$ portion of the file on storage s_i. Here, $f(.)$ is a function of ϵ_i,γ_i,α_2, and α_1. In the case that the reliability of the system is more important for us, the files should be distributed on more reliable storages. Also, x_i for the more reliable storages should be greater. In contrast, in the case that the security is more important, the data should be stored on more secure storages. It should be noted that we just want give an idea of a possible greedy solution. Therefore, we do not discuss the details of function $f(.)$, and we leave it for future work. The hight-level idea of the greedy solution is shown in Algorithm 1.

V. Evaluations

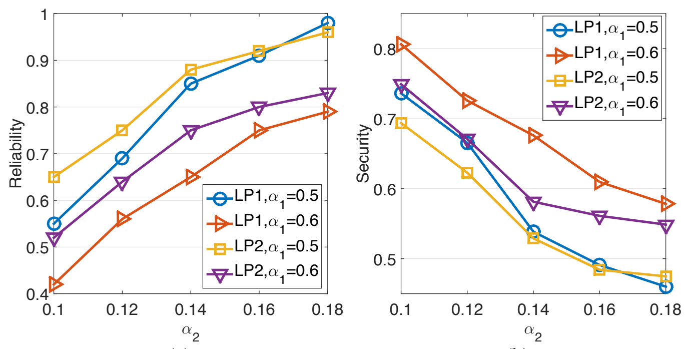
In this section, we evaluate our proposed proposed secure storage system through simulation results. We discuss discuss the setting in the simulations. We then present the simulation results. At the end, we summarize our finding from the simulation runs.

A. Simulation Setting

In order to evaluate our methods, we implemented a simulator in the Matlab environment. In our simulations, we use the Linprog tool of Matlab to find the solution of our proposed liner programming optimization. In order to use Linprog, we need to change the equality constraints to inequality equations. For this purpose, we modify Equation (19) to two equations $z_j \leq \sum_{i:s_i \in R_j}$ and $z_j \geq \sum_{i:s_i \in R_j}$.

We run all of the simulations on 100 random networks. For each network, the reliability and the eavesdropping probability of each storage is selected randomly from a given range, which will be mentioned for each presented plot. We compare the optimal solutions of the mixed integer-linear programming and the two relaxed linear programming optimizations. In the simulations, we measure the effect of the following metrics on reliability and the security of the proposed methods:

- α_1: is the weight that is assigned to the security. The higher values of α_1 means that the security has more important role in the utility. As a result, for a higher α_1, our solutions find a content distribution that results in more secure data storage, which might be less fault tolerance.



(a) (b)

Fig. 3. The effect of α_2 on the reliability and the security of the data storage. $\epsilon_i \in [0, 0.1]$, $\gamma_i \in [0, 0.5]$. (a): measuring the reliability of the system. (b): Measuring the security of the system.

- α_2: is the weight that we assign to the fault tolerance. A high value for α_2 results in more fault-tolerant distributed data storage.
- ϵ: we measure the effect that the failure probability has on the fault tolerance and the security of the storage.
- γ: by changing the range of γ, we evaluate the effect of eavesdropping probability on the security of the system.

In the simulations, we evaluate the reliability and the security of our proposed method in different scenarios.

B. Simulation Result

In the first experiment, we measure the reliability of the data storage. We define the reliability (fault tolerance) as the probability that the non-failed data storages are sufficient to retrieve the original data. For this purpose, the number of stored coded packets on the remaining data storages should be more than or equal to m. In other words, the summation of x_i for the working data storages should be at least 1. We set the failure probability of each data storage to a random value in the range of $[0, 0.1]$. Also, the eavesdropping probability of each data storage (γ)is in a range of $[0, 0.5]$. Figure 3(a) shows the result of our simulations.

In Figure 3(a), we compare the LP1 and LP2 methods in the cases of $\alpha_1 = 0.5$ and $\alpha_1 = 0.6$. The figure shows that the reliability of all of the methods increase as we increase α_1. The reason is that, in the objective function of both of the optimizations, α_1 is the weight of reliability. As a result, when we increase α_1, each data storage stores more coded packets. In this way, if a set of data storages fail, there is a higher chance that the rest of data storages have a sufficient number of coded packets to retrieve the original data. Moreover, Figure 3(a) depicts that the reliability in the case of $\alpha_1 = 0.6$ is less that with $\alpha_1 = 0.5$. The reason is that, a greater α_1 gives more importance to the security of the system. The figure shows that in general, the reliability of the LP2 method is more than that of the LP1 method. It should be noted that the time complexity of the LP2 method is less than that of the LP1 method.

In the next experiment, we measure the security of the data storage system. We define the security of the data storage system as the probability that an eavesdropper can access to

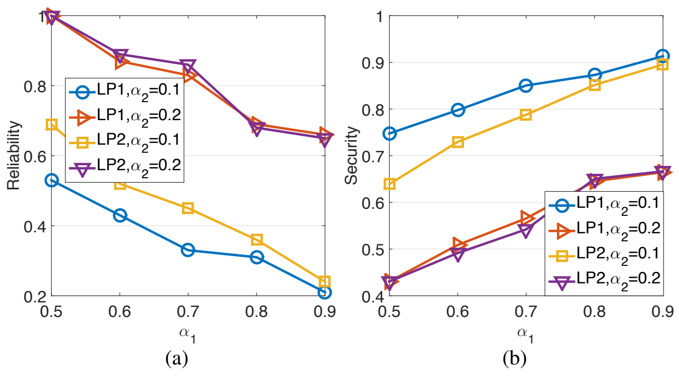


Fig. 4. The effect of α_1 on the reliability and the security of the data storage. $\epsilon_i \in [0, 0.1]$, $\gamma_i \in [0, 0.5]$. (a): measuring the reliability of the system. (b): Measuring the security of the system.

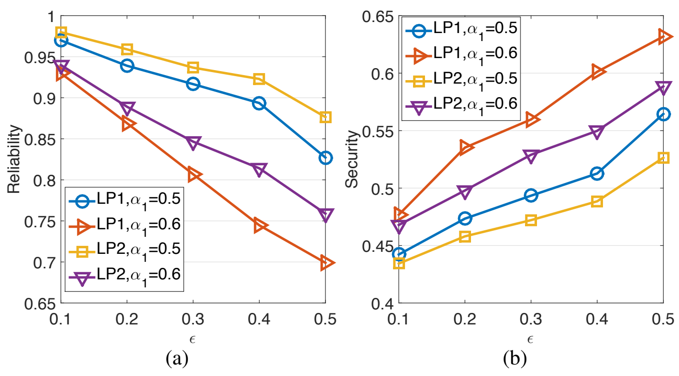


Fig. 5. The effect of ϵ on the reliability and the security of the data storage. $\gamma_i \in [0, 0.5]$. (a): measuring the reliability of the system. (b): Measuring the security of the system.

a set of data storages that have a sufficient number of coded packets to retrieve the original data. In Figure 3(b), $\alpha_1 = 0.5$ and $\alpha_1 = 0.6$. Also, the failure and eavesdropping probability of each storage is selected randomly in the ranges of $[0, 0.1]$ and $[0, 0.5]$, respectively. The figure shows that as we increase α_2, the security of the data storage system decreases. Because, as we increase α_2, the reliability plays a more important role in the objective function. Consequently, in order to minimize the objective function, more coded packets need to be stored on the data storages. In this case, the probability that an eavesdropper can collect a sufficient number of coded packets from the data storages that are accessed increases. Figure 3(b) shows that, the LP1 method provides more security compared to the LP2 method. It can be inferred from Figures 3(a) and (b) that, as the reliability of a data storage system increase, its security decreases. Also, a more secure data storage is less robust against failures.

In the next experiment, we measure the effect of α_1 on the reliability of the data storage system. We select the failure probability of each data storage randomly in the range of $[0, 0.1]$. Also, the eavesdropping probabilities are in the range of $[0, 0.5]$. We change α_1 from 0.5 to 0.9, and show the result in Figure 4(a). The figure depicts that, as we increase α_1, the reliability of the system decreases. This is because, a greater α_1 results in less stored coded packets on the data storages. Consequently, the system will be less resilient to the storage failures, which reduces the system reliability. In the case that $\alpha_2 = 0.2$, the system is more reliable compared to the case with $\alpha_2 = 0.1$. Also, the reliability of the system in the cases of LP1 and LP2 methods are very close.

In Figure 4(b), we change α_1 from 0.5 to 0.9, and measure the security of the system. We select the failure probability of each data storage randomly in the range of $[0, 0.1]$. Also, the eavesdropping probabilities are in the range of $[0, 0.5]$. We run the simulations in the cases of LP1 and LP2 methods with $\alpha_2 = 0.1$ and $\alpha_2 = 0.2$. As we expected, the security of the system increase as we increase α_1. This is because of less stored coded packets on each data stored when α_1 increases. However, as Figure 4(a) shows, the reliability decreases as we increase α_1. Figure 4(b) shows that, the security of the LP1 and LP2 methods are very close to each other. Moreover, $\alpha_2 = 0.1$ results in a more secure data storage.

In order to evaluate the effect of data storage failure probability on the reliability of the system, we change the range of failure probability from $[0, 0.1]$ to $[0, 0.5]$ and measure the reliability of the system. The eavesdropping probability of each data storage is in the range of $[0, 0.5]$. Moreover, α_1 is equal to 0.1. The simulation is run in the cases of $\alpha_1 = 0.5$ and $\alpha_1 = 0.6$ for the LP1 and LP2 methods. The results of the simulations are plotted in Figure 5(a). It is clear that a higher data storage failure should result in a less data storage system, which is confirmed in the figure. The reliability of the LP1 and LP2 methods are very close. In addition, $\alpha_1 = 0.5$ results in a more reliable data storage compared to that of $\alpha_1 = 0.6$.

In our last experiment, we evaluate the effect of data storage failure probability on the security of the system. In Figure 5(a), the eavesdropping probability is selected in the range of $[0, 0.5]$. Also, α_2 is set to 0.2. We change the failure probability of the range of storage probability from $[0, 0.1]$ to $[0, 0.5]$, and measure the reliability of the distributed storage system. The figure shows that there is not huge gap between the reliability of the LP1 and LP2 methods. Moreover, as we increase α_1 from 0.5 to 0.6, the reliability of the LP1 and LP2 methods decrease. For $\alpha_1 - 0.5$, the reliability of the LP1 method is more than 0.98, which is very close to the reliability of the LP2 method.

In the last experiment, we measure the effect that the storage failure probability has on the vulnerability of the system against eavesdropper attacks. We select the eavesdropping probability randomly from the range of $[0, 0.5]$. In addition, α_2 is set to 0.2. We increase the range of storage reliability from $[0, .01]$ to $[0, .05]$, and evaluate the security of the system. It might be strange that the security of the system increase as we increase the failure probability. The reason is that, when we increase the storage failure probability, there are two ways to minimize the objective function. One approach is to increase the redundancy to increase the reliability of the system. The second approach is to reduce the redundancy in order to in enhance the security. Since the failure probability is becoming high, increasing the redundancy has a greater negative effect on the security of the system. Since α_1 is high, increasing the redundancy cannot compensate the negative impact on the security of the system. As a result, the assigned redundancy cannot be hight, which increase the security of the system.

C. Simulation Summary

We can summarize the our finding from the simulation results in this section as follows:

- The reliability of the distributed data storage and its vulnerability have negative correlation. As the reliability of the system increases, its security decreases.
- The reliability and the security of the LP1 and LP2 methods are very close to each other. Specially, in the case that the storage failure probability is low.
- As the failure probability of the storages increase, the security of the system increases as well.

VI. Conclusion

Random linear network coding is a method which is used in wired and wireless networks to increase the throughput of the networks and provide reliable transmissions. Another application of network coding is in distributed storage systems to store a large data on different storages and provide fault tolerance against storage failures. Using random linear network coding, the set of packets can be encoded to infinite number of packets, and a subset of these coded packets are enough to retrieve the original data. In addition to provide fault tolerance and enhancing the reliability of the system, random linear network coding can be used to protect the data from eavesdropper. An unauthorized user is not able to decode the coded packets and retrieve the original data unless it has access to a sufficient number of coded packets.

In this work, we use random linear network coding to design a fault-tolerant and secure data storage. Increasing the redundancy enhances the fault tolerance of a distributed data storage. On the other hand, it makes the system more vulnerable against eavesdropper attacks, since the more is the redundancy the more is the probability that an eavesdropper can access a sufficient number of coded packets. In this work, we perform a trade-off between security of a distributed storage system and its fault tolerance. We formulate the problem as a mixed integer and linear programming, and relax it to linear programming, which can be solved in polynomial time.

Acknowledgment

This work is supported in part by NSF grants CNS 149860, CNS 1461932, CNS 1460971, CNS 1439672, CNS 1301774, ECCS 1231461, ECCS 1128209, and CNS 1138963.

References

[1] P. F. Oliveira, L. Lima, T. T. Vinhoza, J. Barros, and M. Médard, "Trusted storage over untrusted networks," in *IEEE GLOBECOM 2010*, 2010, pp. 1–5.

[2] ——, "Coding for trusted storage in untrusted networks," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1890–1899, 2012.

[3] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[4] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," in *ACM SIGCOMM*, 2006, pp. 243–254.

[5] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *ACM SIGCOMM*, 2007.

[6] P. Ostovari, J. Wu, and A. Khreishah, "Network coding techniques for wireless and sensor networks," in *The Art of Wireless Sensor Networks*, H. M. Ammari, Ed. Springer, 2013.

[7] S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[8] T. Ho, M. Médard, R. Koetter, D. 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.

[9] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782– 795, Oct 2003.

[10] H. Djandji, "An efficient hybrid arq protocol for point-to-multipoint communication and its throughput performance," *IEEE Transactions on Vehicular Technology*, vol. 48, no. 5, pp. 1688–1698, 1999.

[11] B. Zhao and M. Valenti, "The throughput of hybrid-ARQ protocols for the gaussian collision channel," *IEEE Transactions on Information Theory*, vol. 47, no. 5, pp. 1971–1988, 2001.

[12] L. Rizzo and L. Vicisano, "RMDP: an FEC-based reliable multicast protocol for wireless environments," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 2, no. 2, pp. 23–31, 1998.

[13] G. C.Clark and J. B. Cain, *Error-correction coding for digital communications*. Springer, 1981.

[14] S. Lin and D. J. Costello, *Error control coding: Fundamentals and Applications*. Prentice-hall Englewood Cliffs, NJ, 2004.

[15] W. Ryan and S. Lin, *Channel codes: classical and modern*. Cambridge University Press, 2009.

[16] L. Lu, M. Xiao, M. Skoglund, L. Rasmussen, G. Wu, and S. Li, "Efficient network coding for wireless broadcasting," in *IEEE WCNC*, 2010, pp. 1–6.

[17] L. Lu, M. Xiao, and L. Rasmussen, "Relay-aided broadcasting with instantaneously decodable binary network codes," in *ICCCN*, 2011, pp. 1–5.

[18] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 2, pp. 914–925, 2009.

[19] S. L. Howard, C. Schlegel, and K. Iniewski, "Error control coding in low-power wireless sensor networks: When is ecc energy-efficient?" *EURASIP Journal on Wireless Communications and Networking*, vol. 17, no. 2, pp. 29–29, 2006.

[20] M. Vuran and I. F. Akyildiz, "Error control in wireless sensor networks: a cross layer analysis," *ACM Transactions on Networking, IEEE*, vol. 17, no. 4, pp. 1186–1199, 2009.

[21] G. Fox, "Peer-to-peer networks," *Computing in Science & Engineering*, vol. 3, no. 3, pp. 75–77, 2001.

[22] Y. Liu, Y. Guo, and C. Liang, "A survey on peer-to-peer video streaming systems," *Peer-to-peer Networking and Applications*, vol. 1, no. 1, pp. 18–28, 2008.

[23] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer internet video broadcast," *Proceedings of the IEEE*, vol. 96, no. 1, pp. 11–24, 2008.

[24] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *IEEE INFOCOM 2005*, vol. 4, 2005, pp. 2235–2245.

[25] S. Pawar, S. El Rouayheb, H. Zhang, K. Lee, and K. Ramchandran, "Codes for a distributed caching based video-on-demand system," in *IEEE ASILOMAR*, 2011, pp. 1783–1787.

[26] H. Zhang, M. Chen, A. Parekh, and K. Ramchandran, "A distributed multichannel demand-adaptive p2p vod system with optimized caching and neighbor-selection," in *SPIE*, 2011, pp. 81 350X–81 350X.

[27] P. Ostovari, J. Wu, A. Khreishah, and N. B. Shroff, "Scalable video streaming with helper nodes using random linear network coding," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, pp. 1–14, 2015.

[28] P. Ostovari, A. Khreishah, and J. Wu, "Multi-layer video streaming with helper nodes using network coding," in *IEEE MASS*, 2013, pp. 524–532.