CHAPTER 1

TOWARDS A SECURE FRAGMENT ALLOCATION OF FILES IN HETEROGENEOUS DISTRIBUTED SYSTEMS

Yun Tian, PHD[1], Mohammed I. Alghamdi, PHD[2], Xiaojun Ruan, PHD[3], Jiong Xie, PHD[1] and Xiao Qin, PHD[1]

[1]Auburn University, Auburn, Alabama
[2]Al-Baha University, Al-Baha City, Kingdom of Saudi Arabia
[3]West Chester University of Pennsylvania, West Chester, PA

There is a growing demand for large-scale distributed storage systems to support resource sharing and fault tolerance. Although heterogeneity issues of distributed systems have been widely investigated, little attention has been given to security solutions designed for distributed storage systems with heterogeneous vulnerabilities. To address this issue, we design a Secure Fragment Allocation Scheme called S-FAS to improve security of a distributed system where storage sites have a wide variety of vulnerabilities. In the S-FAS approach, we integrate file fragmentation with the secret sharing technique in a distributed storage system with heterogeneous vulnerabilities. Storage sites in a distributed system are categarized into a variety of different server types based on vulnerability characteristics. Given a file and a distributed system, S-FAS allocates fragments of the file to a set of nodes with various vulnerability characteristics. Data confidentiality is preserved because fragments of a file are allocated to multiple types of storage nodes. We built storage assurance and dynamic assurance models to evaluate the quality of the security offered by S-FAS.

We develop a Secure Allocation Processing (SAP) algorithm for the S-FAS scheme to improve the security level and consider its performance using the heterogeneous feature of a large distributed system. We also implement a prototype using the multi-threading technique for S-FAS where the SAP algorithm is incoporated. Analysis results of assurance show that fragment allocations made by S-FAS lead to enhanced security because of the consideration of heterogeneous vulnerabilities in distributed storage systems. Experimental results based on the prototype show that the proposed solution not only improves the security level, but also increases the throughput of the distributed storage system with heterogeneous vulnerabilities. -

1.1 INTRODUCTION

1.1.1 Security Problems in Distributed Systems

An increasing number of scientific and business files need to be stored in large-scale distributed storage systems. The confidentiality of security-sensitive files must be preserved in modern distributed storage systems, because such systems are exposed to an increasing number of attacks from malicious users [17].

Although there exist many security techniques and mechanisms (see, for example, [7] and [11]), it is quite challenging to secure data stored in distributed systems. In general, security mechanisms need to be built for each component in a distributed system, then a secure method of integrating all the components in the system can be implemented. It is critical and important to maintain the confidentiality of files stored in a distributed storage system when malicious programs and users compromise some storage nodes in the system.

In addition to cryptographic systems, secret sharing is an approach to providing data confidentiality by distributing a file among a group of n storage nodes, to each of which a fragment of the file is allocated. The file can be reconstructed only when a sufficient number (e.g., more than k) of the fragments are available to legitimate users. Attackers are unable to reconstruct a file using the compromised fragments when a group of servers are compromised if fewer than k fragments are disclosed.

1.1.2 Heterogeneous Vulnerabilities

In a large-scale distributed system, different storage sites use a variety of methods to protect data. The same security policy may be implemented in various mechanisms. Data encryption schemes may vary; even with the same encryption scheme, key lengths may vary across the distributed system. The above mentioned factors can contribute to different vulnerabilities among storage sites. Although security mechanisms deployed in multiple storage sites can be implemented in a homogeneous way, different vulnerabilities may exist due to heterogeneities in computational units.

We started to address security heterogeneity issues by categorizing storage servers into different server-type groups. Each server type represents a level of security vulnerability. In a server-type group, storage servers with the same vulnerabilities share the same weaknesses that allow attackers to reduce the servers' information assurance. Although it may be difficult to categorize all servers in a system into a large number of groups, a practical way of identifying server types is to organize those with similar vulnerabilities into one group.

In light of server types and heterogeneous vulnerabilities, we investigate in this study a fragment allocation scheme called S-FAS to improve security of a distributed system where storage sites have a wide variety of vulnerabilities.

1.1.3 File Fragmentation and Allocation

The file fragmentation technique is often used in distributed and parallel systems to improve availability and performance. Several file fragmentation schemes have been proposed to achieve high assurance and availability in large distributed systems [1][21]. In real-world distributed systems, the fragmentation technique is usually combined with replication to achieve better performance at the cost of increased security risk to data stored in the systems. A practical distributed system normally contains multiple heterogeneous servers providing services with various vulnerabilities. Unfortunately, the existing fragmentation algorithms do not take the heterogeneity issues into account.

To address the aforementioned limitations, we focused on the development of a file fragmentation and allocation approach to improving the assurance and scalability of a heterogeneous distributed system. If one or more fragments of a file have been compromised, it is still very difficult and time-consuming for a malicious user to reconstruct the file from the compromised fragments. Our solution is different from those previously explored, because it utilizes heterogeneous features regarding vulnerabilities among servers.

To evaluate our method for fragment allocations, we developed static and dynamic assurance models to quantify the assurance of a heterogeneous distributed storage system handling data fragments.

1.1.4 Main Contributions

The following are the main contributions of this study: We addressed the heterogeneous vulnerability issue by dividing storage nodes of a distributed system into different server-type groups based on their vulnerabilities. Each server-type group - represents a group of storage nodes with the same group of security vulnerabilities. We proposed a secure fragmentation allocation scheme called S-FAS to improve security of a distributed system where storage nodes have a wide variety of vulnerabilities. We developed storage assurance and dynamic assurance models to quantify information assurance and to evaluate the proposed S-FAS scheme. We developed a secure allocation processing

(SAP) algorithm to improve security and system performance by considering the heterogeneous feature of a large distributed system. We discovered principles to improve assurance levels of heterogeneous distributed storage systems. The principles are general guidelines to help designers achieve a secure fragment allocation solution for distributed systems. In order to conduct the performance analysis for the S-FAS scheme and SAP allocation algorithm, we developed a prototype for our S-FAS scheme.

1.1.5 The Organization of this Chapter

The rest of this chapter is organized as follows. In Section 1.2, we review related work. Section 1.3 presents the system and threat models of this study. Section 1.4 describes S-FAS–a secure fragmentation allocation scheme. In Section 1.5, we describe static and dynamic assurance models for distributed storage systems. In Section 1.6, the SAP allocation algorithm Principles are presented and the architecture and design of our prototype is illustrated. In Sections 1.7, we quantitatively evaluate the assurance and performance analysis of the proposed S-FAS scheme combined with our proposed SAP algorithm in the context of distributed systems. Section 1.8 summarizes this chapter and outlines our future work.

1.2 RELATED WORK

1.2.1 Security Techniques for Distributed Systems

Much research has been performed concerning the improvement of the security of distributed and high-performance computing systems such as grids. For example, Pourzandi *et al.* proposed a structured security approach that incorporates both distributed authentication and distributed access control mechanisms [17].

Intrusion detection techniques have been widely used to provide basic assurance of security in distributed systems, however, most intrusion detection techniques are inadequate to protect data stored in distributed systems [2]. One of the most effective approaches to improving information assurance in distributed systems is intrusion tolerance [10] [21]. To enhance security assurance, researchers have developed a range of intrusion-tolerant tools and mechanisms. The fragmentation technique summarized below is one of the intrusion tolerance methods that can be used in combination with intrusion detection techniques.

1.2.2 Fragmentation Techniques

A fragmentation technique partitions a security sensitive file into multiple fragments that are distributed across different nodes in a distributed system.

A lot of fragmentation schemes have proven to be valuable tools in the improvement of the security of data stored in distributed systems (for example, [20][5][6][1][8][9]).

Many fragmentation approaches aim to improve availability and performance of distributed systems by applying data replication methods. For example, Dabek *et al.* developed a wide-area cooperative storage system in which they implemented a fragmentation scheme to improve availability and to facilitate load balancing [3].

Although a scheme combining fragmentation and replication can enhance performance and availability, data replications may impose security risks due to an increasing number of file fragments handled by distributed storage servers. A file is more likely to be compromised when more replications of the file are stored in a distributed storage system.

All existing file fragmentation technologies are inadequate to address the issue of heterogeneous vulnerabilities in large-scale distributed systems. Our preliminary results show that security can be improved in a distributed storage system when a fragmentation scheme incorporates the heterogeneous-vulnerability feature.

1.2.3 Secret Sharing

Secret sharing is a method of distributing a secret among a group of participants, by allocating each a share of the secret. The secret can be successfully reconstructed only when a sufficient number of shares are combined together [16][9]. Shamir proposed the (k, n) secret sharing scheme that divides data D into n pieces in such a way that D can be easily reconstructed from any k pieces. If fewer than k pieces are disclosed, the data cannot be reconstructed from the revealed pieces. The secret sharing scheme has been extended and employed in different application domains [19]. For example, Bigrigg *et al.* proposed an architecture called PASIS for secure storage systems. The PASIS architecture integrates the secret sharing scheme with information dispersal to improve security, integrity and availability [4][12]. In a storage system with PASIS, the confidentiality of stored data in the system is still preserved, even if an attacker compromises a limited (i.e., fewer than the threshold) subset of storage nodes. The aforementioned secret-sharing solutions designed for distributed storage systems ignore the issue of heterogeneous vulnerabilities. This fact motivated us to extend the secret sharing scheme by considering heterogeneity in vulnerabilities, in the context of distributed storage systems.

1.2.4 Comparison of Our Work with Existing Solutions

The fragment allocation solution we describe in this chapter is entirely different from the existing fragment allocation schemes found in the literature. Our approach incorporates the vulnerability heterogeneity feature of distributed storage systems into file fragment allocation. Our solution captures hetero-

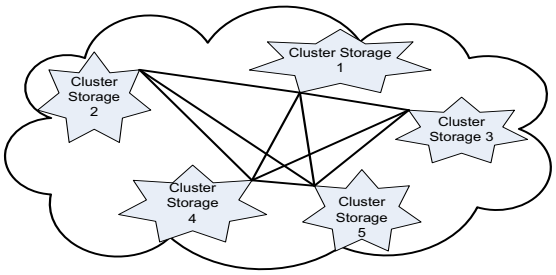


Figure 1.1: A distributed storage system is comprised of a set of cluster storage subsystems. Multiple fragments of a file can be stored either in storage nodes within a single cluster storage subsystem or in nodes across multiple cluster storage subsystems.

geneous features regarding vulnerabilities of the nodes in order to improve the security level of the data stored in a distributed system. In this study, the data replication technique is not considered, because fragment allocation and data replication are independent of each other. Thus, the availability and performance of fragment allocation schemes can be improved when data replication modules are integrated.

1.3 SYSTEM AND THREAT MODELS

In this section, we will outline the system and threat models that capture main characteristics of distributed storage systems. The system model is used as a basis to design the S-FAS fragmentation allocation scheme, whereas the threat model helps us identify vulnerabilities and certain potential attacks in distributed storage systems.

1.3.1 System Model

The S-FAS fragmentation allocation scheme was designed for a distributed storage system (see Fig. 1.1) where each storage site is a cluster storage subsystem. Different cluster storage subsystems may be connected within some subnetworks to form a larger scale distributed storage sysytem. A cluster storage subsystem consists of a number of storage nodes and a gateway. Considering heterogeneous vulnerability in large-scale storage systems, we divide storage nodes into different server-type groups.

Before presenting details on the system model, let us summarize all notations used throughout this Chapter in Table 1.1.

Table 1.1: Notations

Items	Meaning	Items	Meaning
U	The whole system considered	N	Total Number of storage nodes
F	A file stored in the system	F_i	Fragment i of file F
K	Total number of storage node types	m	Threshold of secret sharing scheme
T_j	Server type j in the system	$P(X)$	Probability of event X occurring
R_i	Cluster storage subsystem	$P(Y)$	Probability of event Y occurring
r_{ij}	Node j in subsystem i	$P(Z)$	Probability of event Z occurring
α	An allocation mapping of file F	$P(V)$	Probability of event V occurring
L	Number of subsystems in the whole system	H_i	Number of server nodes in the subsystem i
Y	An event if X occurs, at least m fragments can be compromised using the same attack method	n	Total number of fragments of each file in the secret sharing scheme
X	The event by which a set of storage nodes is chosen to be attacked	Z	The event of a successful attack to a certain fragment of a file
S_j	The size of a certain server type j in a cluster	P_N	The probability of a successful attack on a node
P_f	The probability of successfully compromising a fragment in a compromised node	$SA(\alpha)$	The storage assurance of an allocation mapping α of file F
q	Number of fragments needed to reconstruct a file transmitted from outside of the subsystem	g	Number of fragments compromised out of the q fragments transmitted across subsystems
P_L	The probability that a fragment is intercepted during its transmission	P_D	The probability that a file F is intercepted because of the compromised transmitted fragments
V	The event file F is compromised under one attack method	DA(α)	The dynamic assurance of an allocation mapping α of file F
I_{ij}	Decreasing sorted list of fragments to be allocated	LoadB	work load that a fragment brings to the node where it is stored

In this study, we will consider a distributed storage system containing L cluster storage subsystems (i.e., $R_1, R_2, ..., R_L$). Cluster storage subsystem R_i consists of H_i storage nodes (i.e., $R_i = \{r_{i1}, r_{i2}, ..., r_{iH_i}\}$). All the storage nodes connected in cluster R_i have heterogeneous vulnerabilities.

Since all the nodes, including a master node, are fully connected in a cluster storage subsystem, we model the topology of a cluster storage system as a general graph. Cluster storage subsystem R_i has a gateway, which hides the cluster's internal architecture from users by forwarding file requests to storage nodes.

Data in cluster storage subsystem R_i can be accessed through its master node. When a read request is submitted to cluster R_i, the master node is responsible for reconstructing file fragments and returning the file to users. When a write request of a file is issued, the master node updates all fragments of the file.

Legitimate users access cluster storage subsystems through master nodes; malicious users may bypass the master nodes to access storage nodes without being authorized. See Section 1.3.2 below for details on the threat model.

1.3.2 Threat Model

It is not reasonable to assume that if a malicious user breaks into a storage node, fragments of a file stored on the node are thereby compromised. Normally, a malicious user requires two steps to compromise fragments of a file stored on a server. First, the malicious user must successfully compromise the server. Second, fragments are retrieved by the malicious user.

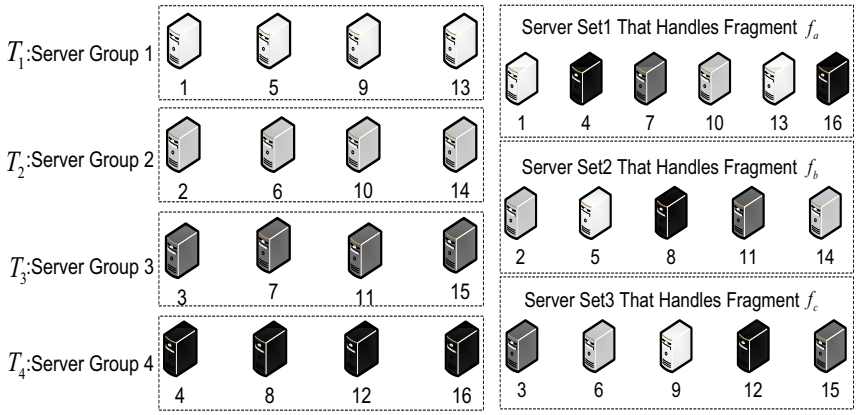
Let P_N be the probability that a storage server is successfully attacked; let P_f be the probability that authorized users retrieve fragments stored on the server, provided that the server has been compromised. We define event Z as a successful attack on a fragment (i.e., unauthorized disclosure of the fragment). Since the above two consecutive attack steps are independent, the probability that event Z occurs is a product of probability P_N and probability P_f. Thus, the probability that a fragment is disclosed to an unauthorized attacker can be expressed as:

$$\mathrm{P}\,(\mathrm{Z}) = P_N * P_f. \tag{1.1}$$

Given two storage nodes with different vulnerabilities, successful attacks of the nodes are not correlated. This statement is true for many potential threats, because compromising one storage node does not necessarily lead to the successful attack of another.

1.4 S-FAS: A SECURE FRAGMENT ALLOCATION SCHEME

In this section, we first outline the motivation for addressing the heterogeneity issues in the vulnerability of distributed storage systems. Next, we describe a



(a) A distributed storage system contains 16 storage nodes, which are divided into 4 server-type groups, i.e., T_1, T_2, T_3, and T_4.

(b) Possible insecure file fragment allocation decision made using a hashing function (see Eq. 11 in [1])

Figure 1.2: A Motivational Example

security problem addressed in this study. Last, we present a secure fragment allocation scheme called S-FAS for distributed storage systems.

1.4.1 Heterogeneity in the Vulnerability of Data Storage

Since the existing security techniques (see Section 1.2) developed for distributed systems are inadequate for distributed systems with heterogeneity in vulnerabilities, the focus of this study is heterogeneous vulnerabilities in large-scale distributed storage systems. Vulnerabilities of storage nodes in a distributed system are heterogeneous in nature due to the following four main reasons. First, storage nodes have different ways to protect data. Second, a security policy can be implemented in a variety of mechanisms. Third, the key length of an encryption scheme may vary across multiple storage nodes. Fourth, heterogeneities exist in computational units of storage sites. We believe that future security mechanisms for distributed systems must be aware of vulnerability heterogeneities.

1.4.2 A Motivational Example

If the above heterogeneous vulnerability features are not incorporated into fragment allocation schemes for distributed storage systems, a seemingly secure fragment allocation decision can lead to a breach of data confidentiality. The following motivational example illustrates a security problem caused by ignoring vulnerability heterogeneities.

Let us consider a file F with three partitioned fragments: f_a, f_b, and f_c, and a distributed storage system (see Fig. 1.2(a)) that contains 16 storage nodes divided into 4 server-type groups (or server groups for short), i.e., T_1, T_2, T_3, and T_4. Storage nodes in each server group offer similar services with the same vulnerabilities. In this example, server group T_1 consists of nodes r_1, r_5, r_9, r_{13}, i.e., $T_1 = \{r_1, r_5, r_9, r_{13}\}$. Similarly, we define the other three server groups as: $T_2 = \{r_2, r_6, r_{10}, r_{14}\}$, $T_3 = \{r_3, r_7, r_{11}, r_{15}\}$, and $T_4 = \{r_4, r_8, r_{12}, r_{16}\}$.

Fig. 1.2(b) shows that it is possible to make insecure fragment allocation decisions when vulnerability heterogeneity is not taken into account. The decision made using a hashing function (see Eq. 11 in [1]) randomly allocates the three fragments of file F to three different nodes, each of which belongs to one of the three server sets illustrated in Fig. 1.2(b). For example, the three fragments f_a, f_b, and f_c are stored on nodes r_1, r_6, and r_8, respectively. This fragment allocation happens to be a good solution, because r_1, r_6, and r_8 have different vulnerabilities as the three nodes belong to different server groups (i.e., T_1, T_2, and T_4). A malicious user must launch at least three successful attacks (one for each server group) in order to compromise all three fragments.

The above fragment allocation scheme fails to address the threat described in Section 1.3. This is because an attacker can first retrieve one fragment of F by compromising a single node, and then wait for the other two fragments to be passed through the compromised node. To solve this security problem, Zanin *et al.* developed a static algorithm to decide whether a particular storage node is authorized to handle a file fragment of F [12]. Zanin's algorithm can generate an insecure fragment allocation because heterogeneous vulnerabilities are not considered. For example, if the three fragments are respectively stored on nodes r_4, r_8, and r_{12}, which share the same vulnerability in server group T_4 (see Fig. 1.2(b)), rather than three attacks, one successful attack against server group T_4 allows unauthorized users to access the three fragments of file F. Two other insecure fragment allocations are: (1) allocating f_a, f_b, f_c to nodes r_1, r_5, and r_9, respectively; and (2) allocating f_a, f_b, f_c to nodes r_7, r_{11} and r_{15}, respectively. These three fragment allocation decisions are unacceptable, because the fragments are assigned to a group of storage nodes with the same vulnerability, meaning that an attacker who comprised one node within a group can easily compromise the other nodes in the group. The attacker can reconstruct F from f_a, f_b, and f_c stored on the comprised server group.

1.4.3 Design of the S-FAS Scheme

To solve the above security problem, we must incorporate vulnerability heterogeneities into fragment allocation schemes. Specifically, we designed a simple yet efficient approach to allocating fragments of a file to storage nodes with various vulnerabilities. Since allocating fragments of a file into different stor-

age subsystems can degrade performance, our S-FAS scheme attempts to allocate fragments to storage nodes in as few clusters as possible. To improve the assurance of a distributed storage system while maintaining high I/O performance, each cluster storage subsystem must be built with high vulnerability heterogeneity. This causes the fragments of a file to be less likely distributed across multiple storage clusters.

Because of the following two reasons, the S-FAS scheme can significantly improve data security when fragments are stored in a large-scale distributed storage system. First, S-FAS integrates the fragmentation technique with secret sharing. Second, S-FAS addresses the issue of heterogeneous vulnerabilities when file fragments are allocated to a distributed storage system.

The S-FAS scheme makes fragment allocation decisions by following the four policies below:

- **Policy 1:** All the storage nodes in a distributed storage system are classified into multiple server-type groups based on their various vulnerabilities. Each server group consists of storage nodes with the same vulnerabilities.

- **Policy 2:** To improve security of a distributed storage system, S-FAS allocates fragments of a file to storage nodes belonging to as many different server groups as possible. In doing so, it is impossible to compromise the file's fragments using a single successful attack method.

- **Policy 3:** The fragments of a file try to be allocated to nodes within a wide range of vulnerabilities all within the fewest cluster storage subsystems which are close to clients. The goal of this policy is to improve performance of the storage system by making the fragments less likely to be distributed across too many distant clusters.

- **Policy 4:** The (m, n) secret sharing scheme is integrated with the S-FAS allocation mechanism.

If a file's fragment-allocation decisions are guided by the above four policies, successful attacks against less than m server groups have little chance of gaining unauthorized accesses to files stored in a distributed system. In other words, if the number of compromised fragments of a file is less than m, attackers are unable to reconstruct the file from the fragments that are accessed by the unauthorized attackers. The S-FAS scheme can improve information assurance of files stored in a distributed storage system without enhancing confidentiality services deployed in cluster storage subsystems of the distributed system, because S-FAS is orthogonal to security mechanisms that provide confidentiality for each server group in a distributed storage system. Thus, S-FAS can be seamlessly integrated with any confidentiality service employed in distributed storage systems in order to offer enhanced security services.

1.5 ASSURANCE MODELS

We developed assurance models to quantitatively evaluate the security of a heterogeneous distributed storage system in which S-FAS handles fragment allocations.

1.5.1 Storage Assurance Model

For encrypted files, their encryption keys are partitioned and allocated using the same strategy that handle file fragments. Once a storage node in set U is compromised, file fragments and encryption key fragments stored on the node are both breached. If a malicious user wants to crack a file, at least m nodes within U must be successfully hacked.

We first investigate the probability that a file is compromised using one attack method. Let X be the event that a set of storage nodes is chosen to be attacked. Let Y be the event that if X occurs, at least m fragments can be compromised using the same attack method. As we already defined in Section 1.3, event Z represents a successful attack to a certain fragment of a file. Applying the multiplication principle, we calculate the probability that V - an event that file F is compromised under one attack - occurs as:

$$\mathrm{P(V)} = \sum_{\mathrm{j=1}}^{\mathrm{k}} \mathrm{P(X)P(Y)P(Z)} \tag{1.2}$$

where $P(X)$, $P(Y)$ and $P(Z)$ are probabilities that events X, Y and Z occur when the total number of different server-type groups (server group for short) is K. The probability $P(V)$ is proportional to probability $P(Z)$, which largely depends on the quality of security mechanisms deployed in the storage system, as well as the attacking skills of hackers.

Note that when k equals 1, there is no vulnerability difference among storage nodes. Supposing that all the fragments of a file can be compromised using one successful attack method, the probability that Y occurs becomes 1. Then, we can express $P(V)$ as:

$$\mathrm{P(V)} = \sum_{\mathrm{j=1}}^{\mathrm{k}} \mathrm{P(X)P(Z)} \tag{1.3}$$

Let S_j be the number of storage nodes in *server type* T_j set and N be the total number of nodes in a distributed system. The probability that nodes in set T_j are randomly attacked can be derived as $\mathrm{P(X)} = \frac{\mathrm{S}_j}{\mathrm{N}}$.

Probability $P(Y)$ in Eq. 1.2 can be calculated as follows:

$$\mathrm{P\,(Y)} = \sum_{\mathrm{i=m}}^{\mathrm{n}} \frac{C_{S_j}^{i} C_{N-S_j}^{n-i}}{C_N^n}, (j = 2, \ldots K) \tag{1.4}$$

where C_N^n is the total number of possibilities of allocating fragments of a file, and the product of $C_{S_j}^i$ and $C_{N-S_j}^{n-i}$ is the number of possibilities that a file is compromised using a successful attack method which means at least m (It may be $m+1$, $m+2$, ..., n) fragments of the file are compromised.

To simplify the model, one may assume that security mechanisms and attacking skills have no significant impacts on information assurance of the entire distributed storage system. This assumption is reasonable because of two factors. First, S-FAS is independent of security mechanisms that provide confidentiality for server groups in a distributed storage system. Second, if empirical studies can provide values for probability $P(Z)$, the probability $P(V)$ can be derived from $P(Z)$ and the model (see Eq. 1.4) that calculates $P(Y)$. Since the study of the distribution of $P(Z)$ is not within the range of this work, in Section 1.7 the impact of probability $P(Z)$ on $P(V)$ is ignored by setting the value of $P(Z)$ to 1. Now we can derive Eq. 1.5 from Eq. 1.4 as below:

$$\mathrm{P(V)} = \sum_{\mathrm{j=1}}^{\mathrm{K}} \left(\frac{\mathrm{S_j}}{\mathrm{N}} \mathrm{P(Z)} \sum_{\mathrm{i=m}}^{\mathrm{n}} \frac{\mathrm{C_{S_j}^i C_{N-S_j}^{n-i}}}{\mathrm{C_N^n}} \right) \tag{1.5}$$

The confidentiality of file F is assured if F is not compromised. Thus, we can derive the assurance $SA(\alpha)$ of the storage system from Eq. 1.5 as:

$$\mathrm{SA}\,(\alpha) = 1 - \mathrm{P}\,(\mathrm{V}) = 1 - \sum_{j=1}^{\mathrm{K}} \left(\frac{\mathrm{S}_j}{\mathrm{N}} \mathrm{P(Z)} \sum_{\mathrm{i=m}}^{\mathrm{n}} \frac{\mathrm{C_{S_j}^i C_{N-S_j}^{n-i}}}{\mathrm{C_N^n}} \right) \tag{1.6}$$

1.5.2 Dynamic Assurance Model.

During read and write operations, some fragments of a file may be transmitted among different storage clusters or subnetworks. We assume that data transmissions within a cluster are secure, while connections among clusters and subnetworks may be insecure. Let P_L be the probability that a fragment is intercepted during its transmission on an insecure link. We consider a common case in which some fragments of file F are allocated outside a cluster. The probability P_D that a fragment of F is intercepted during its transmission can be expressed as:

$$\mathrm{P_D} = \mu_1 \mu_2 \mathrm{P_L} + \mu_3 \left[1 - \mathrm{P_L}\right] \mathrm{P_L} \tag{1.7}$$

where $\mu_1 = 1$ indicates that connections among storage clusters are insecure and $\mu_1 = 0$ means the connections are secure. $\mu_2 = 1$ indicates that fragments are transferred among different clusters, otherwise $\mu_2 = 0$. Similarly, $\mu_3 = 1$ means that fragments are transmitted across different subnetworks. When μ_1, μ_2, and μ_3 equal to 0, there is no fragment transmission risk. If q fragments need to be collected outside a cluster processing read/write operations, then probability $\mathrm{P_q(g)}$ that g out of q fragments are intercepted can be expressed as:

$$\mathrm{P_q\,(g)} = \mathrm{C_q^g P_D{}^g (1 - P_D)^{q-g}} \tag{1.8}$$

Now we model the dynamic assurance of an allocation mapping α of file F. For simplicity, let us focus on a time period during which there is only one attempt to attack storage nodes where F is stored. During this time period, we assume that only one read or write operation is issued to access F. There are two cases in which file F can be compromised. First, a malicious user can reconstruct F from m compromised fragments using the same attack method. Second, although less than m fragments are compromised, other g fragments are intercepted during their transmissions. Hence, we can derive the dynamic assurance DA(α)(1.9)from the storage risk (see Eq. 1.5) and the transmission risk (see Eq. 1.8), as shown here:

$$\mathrm{DA}(\alpha) = 1 - \left(\mathrm{P(V)} + \left(\sum_{\mathrm{g=(m-i)}}^{\mathrm{q}} \mathrm{P_q(g)} \right) \sum_{j=1}^{K} \left(\frac{S_j}{N} \sum_{\mathrm{i=0}}^{\mathrm{m-1}} \frac{C_{S_j}^{i} C_{N-S_j}^{n-i}}{C_N^n} \right) \right) \tag{1.9}$$

1.6 SAP ALLOCATION PRINCIPLES AND PROTOTYPE

We developed a prototype using the multi-threading technique for the SAP algorithm to guide the file allocation. We then conducted some experiments to evaluate the performance of SAP integrated in the S-FAS scheme. Results show that the proposed solution can not only improve the storage security level, but also enhance the throughput of the distributed storage system with heterogeneous vulnerabilities.

1.6.1 Allocation Principles

The design of our S-FAS scheme is mainly focused on the improvement of system security. Considering the heterogeneity of the system can also be used to improve system performance, we developed a secure allocating processing (SAP) algorithm for the S-FAS scheme [13] to improve the security level and consider its performance using the heterogeneous feature of a large distributed system. The SAP allocation algorithm, a key compoment, addresses the issues of load balancing, delayed effects caused by the workload variance of many consecutive requests, and the heterogeneous feature of distributed storage systems.

There are three main factors affecting the processing delay of a request from a client, namely, workload, network traffic, and IO latency at storage nodes. In our algorithm design, we consider load balacing among storage nodes and network interconnects to improve the system performance. Initially, all nodes are logically categorized into different types based on the similarities of their vulnerabilities. The basic principles of the SAP allocation algorithm are described below.

1. To sort the two-dimensional list I_{ij} of file(i) fragments(j) in a decreasing order based on the burdens of the input files.
2. To sort all storage nodes in a two-dimensional list in a decreasing order based on their vulnerability type and processing speed.
3. To allocate each Fragment in list I_{ij} to storage nodes.

There are two constraints used when deciding whether a fragment should be allocated to a node:

- There is enough space is enough to store the fragment;
- The available LoadB is higher than 80% (The upper bound of the storage load can be adjusted to different level to control the storage space efficiency on a server) of the LoadB of the current fragment.

It is possible that fragments of a file can be allocated in the same type of nodes. Nevertheless, the chances of allocating multiple fragments of a file to nodes of the same type is much less than those in random allocation scheme.

1.6.2 Prototype Architecture and Design

In order to evaluate the S-FAS scheme along with the SAP algorithm, we designed and implemented a prototype to test system throughput on a distributed storage testbed. We applied the multi-threading technique, allowing one thread to process one request of fragment readwrite to enhance processing performance.

In the implimentation of the prototype, system nodes are divided into three main groups - computing nodes (clients), storage nodes, and a storage server. The storage server is responsible for handling file allocation using the SAP algorithm, managing the metadata and dealing with the incoming read or write requests from the clients. The storage nodes are categorized into a few groups based on their heterogeneous vulnerability features. The client nodes can directly access storage servers through the network interconnect.

1.7 EVALUATION OF SYSTEM ASSURANCE AND PERFORMANCE

The assurance models described in Section 1.5 indicate that system assurance is affected by the number K of storage types, the number N of storage nodes in the system, and the number S_j of nodes of the jth storage type. In addition, threshold m and the number n fragments of a file also have an impact on system assurance.

In this section, we will first evaluate the impact of these factors (K, N, S_j, n and m) on storage assurance (see Sections 1.7.1 to 1.7.5). We will compare our approach with a traditional fragment allocation scheme that does not consider

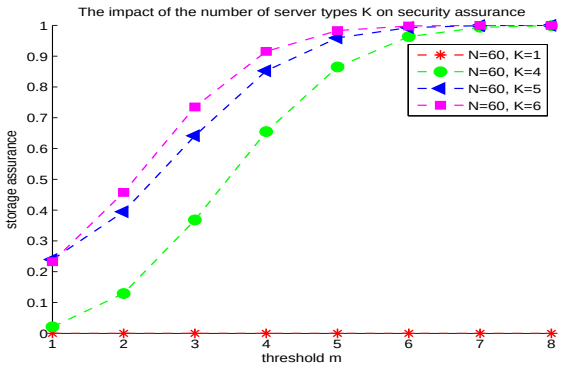


Figure 1.3: Heterogeneous system and homogeneous system using secret sharing scheme.

vulnerability heterogeneities. Then, we will measure dynamic assurance of S-FAS to analyze the impact of P_L and q on system dynamic assurance (see Sections 1.7.6 and 1.7.7). We evaluated a distributed storage system with the threshold value m. The default number n of fragments of a file is set to 12 (the value can vary according to system sizes) and S_j=$\frac{N}{K}$ for all j from 1 to K.

In the last part of this section, we discuss experimental results for the SAP algorithm and the prototype. To explore the affecting factors that influence the system performance, we evaluate the impact of the file size and fragment number on system throughput(see section 1.7.8).

1.7.1 Impact of Heterogeneity on Storage Assurance

If all storage nodes in the evaluated distributed system are identical in terms of vulnerability, the probability that fragments of a file can be compromised using one successful attack method is 1. Fig. 1.3 shows the impact of the number K of storage types on system assurance. The results plotted in Fig. 1.3 suggest that for a distributed system with homogeneous vulnerability, threshold m has no impact on system assurance. For a distributed system with heterogeneous vulnerabilities, the system assurance increases significantly with increasing values of K and threshold m (see Fig. 1.3). Such a trend implies that a high heterogeneity level of vulnerability gives rise to high confidentiality assurance.

1.7.2 Impact of System Size on Storage Assurance

To quantify the impact of system size N on assurance of a file stored in the system, we gradually increase system size from 45 to 70 by increments

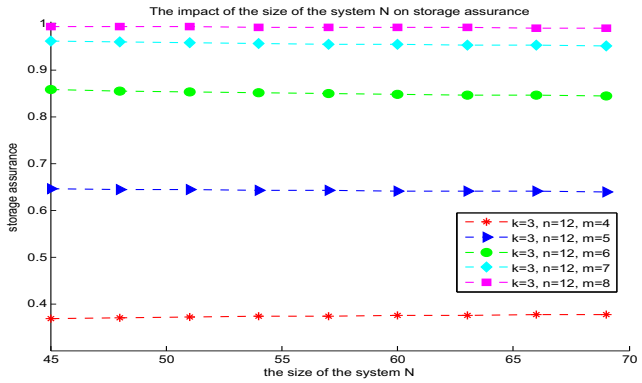


Figure 1.4: The impact of the system size N on storage assurance.

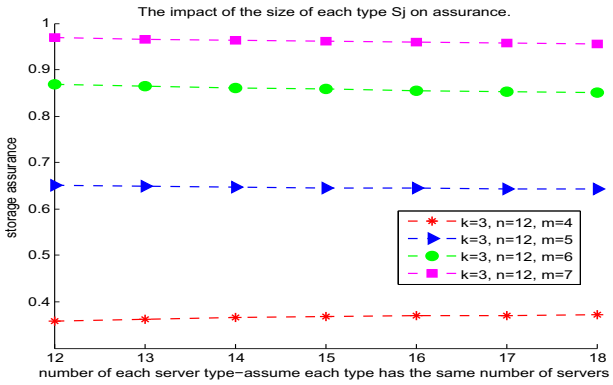


Figure 1.5: The impact of server-group size on data storage assurance. The server-group size means the number of storage nodes in a server-type group.

of 5. We keep k at 3 and also vary m from 4 to 8. Fig. 1.4 reveals that the storage assurance of the system is not very sensitive to the system size, indicating that storage assurance largely depends on the vulnerability heterogeneity level rather than system size. Thus, large-scale distributed storage systems with low levels of vulnerability heterogeneities may have lower assurance than small-scale distributed systems. These results suggest that one way to improve system assurance is to increase vulnerability heterogeneity while increasing the scale of a distributed storage system. A high heterogeneity level in vulnerability helps in increasing threshold m, making it difficult for attackers to compromise multiple server groups and to reconstruct files.

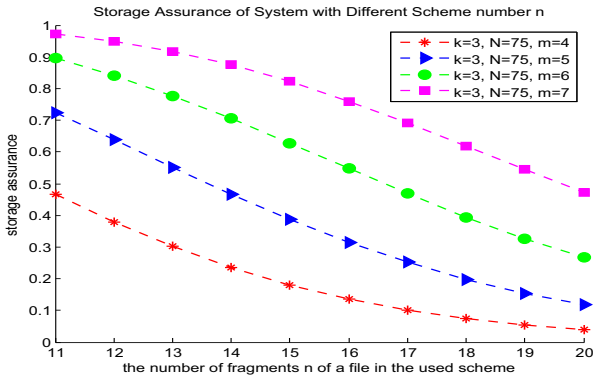


Figure 1.6: The impact of the number n of fragments of a file on storage assurance.

1.7.3 Impact of Size of Server Groups on Storage Assurance

Fig. 1.5 illustrates the impact of server-group size on data storage assurance. Note that the server-group size is the number of storage nodes in a server-type group, in which all the storage nodes share the same group of vulnerabilities. We vary the server-group size from 12 to 18 by increments of 1. We observe from Fig. 1.5 that when threshold m is small (e.g, $m = 4$), the assurance of systems with large server-group sizes is slightly higher than that of systems with small server-group sizes. Interestingly, the opposite is true when the threshold m is large (e.g, $m > 4$). Given a fixed number of storage nodes in a distributed storage system, increasing the server-group size can decrease the number of server groups, which in turn tends to reduce vulnerability heterogeneity. The results shown in Fig. 1.5 match the results in the previous experiments in which a low level of vulnerability heterogeneity results in degraded storage assurance.

1.7.4 Impact of Number n of File Fragments on Storage Assurance

Fig. 1.6 illustrates the impact of the number n of fragments of a file on storage assurance. In this experiment, we increased the number n of fragments from 11 to 20 and measured data storage assurance using our model. The parameters k and N were set to 3 and 75, respectively. We also varied threshold m from 4 to 7. Results depicted in Fig. 1.6 confirm that the system assurance is reduced with the increasing value of fragment number n. The results indicate that a large number of file fragments leads to low data storage assurance of the file. This assurance trend is reasonable because more fragments are likely to be allocated to storage nodes with the same vulnerability. If one storage node is compromised by an attacker, fragments stored on nodes with the same

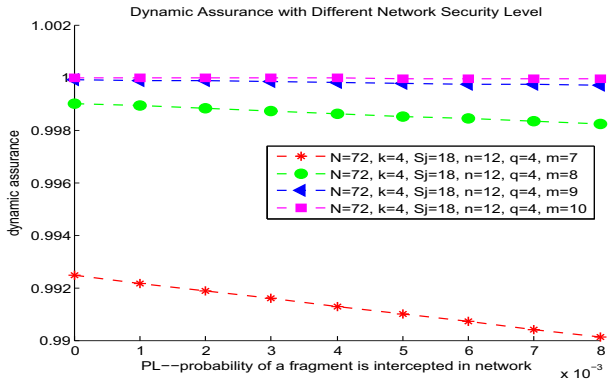


Figure 1.7: Impact of P_L(the probability that a fragment might be intercepted by an attacker during the fragment's transmission.)

vulnerabilities can also be obtained by the attacker, who is therefore more likely to be able to reconstruct the file from the disclosed fragments.

In addition, Fig. 1.6 shows that increasing the value of threshold m can improve storage assurance. This pattern is consistent with the results obtained in the previous experiments.

1.7.5 Impact of Threshold m on Storage Assurance

Figs. 1.3-1.6 clearly show the impact of threshold m on storage assurance of a distributed system. More specifically, regardless of other system parameters, the storage assurance always goes up with the increasing threshold value m. The results indicate that the more fragments an attacker needs in order to reconstruct a file, the higher data storage assurance can be preserved for the file in distributed storage systems. These results suggest that to improve data storage assurance of a file, one needs to partition the file and allocate fragments in such a way that an attacker must compromise more server groups (the best case is m server groups when all fragments of the file are allocated to nodes of different server types) in order to reconstruct the file.

1.7.6 Impact of P_L on Dynamic Assurance

Now we are in a position to evaluate dynamic assurance of distributed storage systems. The three parameters μ_1, μ_2, and μ_3 in Eq. 1.7 have an important impact on dynamic assurance because they indicate whether there is risk during fragment transmission. Please refer to Sections 1.7.1 to 1.7.5 for details on the impacts of a set of parameters on data storage assurance. P_L - the probability that a fragment might be intercepted by an attacker during the fragment's transmission through an insecure link - has a noticeable impact on

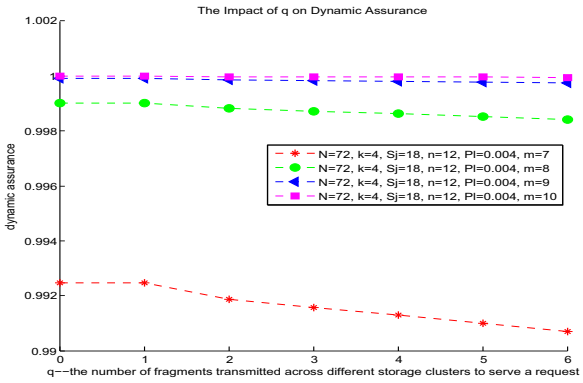


Figure 1.8: Impact of q(q fragments transmitted across storage clusters.)

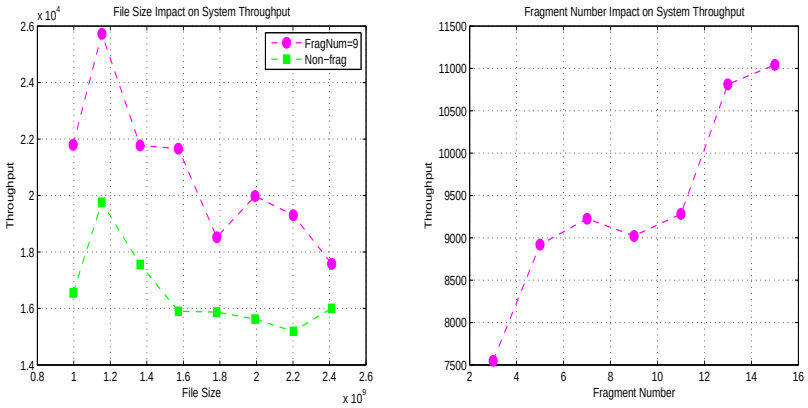
dynamic assurance of a distributed storage system provided that threshold m is small (e.g., smaller than 9). Fig. 1.7 shows the dynamic assurance of a distributed system when P_L is varied from 0 to $8 * 10^{-3}$ by increments of $1 * 10^{-3}$. We also vary threshold m (i.e., m is varied from 7 to 10) to evaluate the sensitivity of dynamic assurance on parameter P_L under different threshold m.

Fig. 1.7 demonstratively confirms that when threshold m is equal to or smaller than 8, a large value of P_L results in low dynamic assurance of the system. These results are expected since a high value of P_L means that the transmitted fragments are likely to be intercepted by an attacker. Once the attacker has collected enough fragments of a security-sensitive file, the file can be reconstructed. When threshold m is larger than 8, the dynamic assurance is not noticeably sensitive to the probability P_L that a fragment is compromised during its network transfer.

1.7.7 Impact of q on Dynamic Assurance

Like parameter P_L, the number q of fragments transmitted to and from a storage cluster also has an impact on the dynamic assurance of a distributed storage system. Intuitively, Fig. 1.8 shows that when the number of fragments of a file that must be transmitted through insecure links is increasing, the dynamic assurance of the file drops. Interestingly, when threshold m is larger than 8, the dynamic assurance becomes very insensitive to the number q of fragments. This observation suggests that when the threshold is small, the S-FAS fragment allocation scheme must specifically attempt to lower the value of q in order to maintain a high dynamic assurance level.

In addition, we observe from Fig. 1.8 that dynamic assurance is always lower than the corresponding storage assurance (where q=0 in Fig. 1.8). This trend is always true because in a dynamic environment, file fragments have



(a) Impact of File Size(file size: byte; throughput: byte/millisecond)

(b) Impact of Fragment Number(Fragment Number: 3, 5, 7, ..., 15; throughput: byte/millisecond)

Figure 1.9: File Size and Fragment Number Impact on System Throughput

to be transmitted through insecure network links where malicious users may intercept the fragments in order to reconstruct files.

1.7.8 Performance Evaluation of S-FAS

We made use of the prototype to evaluate the performance of S-FAS. We conducted some experiments by varying two parameters which noticebly affect both security and performance. We conducted an experiment to test the file allocation process by varying the workload and fragment number of the S-FAS scheme as described in the following.

- File size: In our experiments, the test file size is varied from 950 MB(996147200 bytes) to 2300MB(2411724800 bytes). The details can be seen in the corresponding figures.
- Fragment number of a file: We varied the fragment number from 3 to 15. We set the upper bound to less than 16, since we have 16 storage nodes in our test bed, and the S-FAS performs better when one storage node stores one fragment of a file at most.

Fig. 1.9(a) plots the impact of the allocating file size on the throughput of S-FAS. From Fig. 1.9(a), we observe that the S-FAS scheme improves the throughput compared with the traditional non-fragmented storage method. Because we use the multi-threading method to deal with the fragments of a file in the S-FAS scheme, the system performance is significantly improved in our scheme. The other trend we can observe from Fig. 1.9(a) is that with

the increasing of file sizes, the system throughput firstly increases and then decreases after the file size is larger than 1.2GB. This trend applies to S-FAS and the non-S-FAS scheme. When file size is relatively small, the throughput increases with the increasing of the file size. The reason is that the load is still below the upper bound of the system peak performance. This experiment for high-performance distributed storage systems suggests that implementing the S-FAS scheme using multi-threading method is very important.

Fig. 1.9(b) shows the impact of the fragment number on the throughput. Fig. 1.9(b) shows that the throughput goes up when we increase the fragment number used in the S-FAS scheme. Because we use one thread to deal with one fragment of a file in the S-FAS scheme, the performance is improved with the increasing number of the fragments. The throughput does not noticeably change when the number of fragments is anywhere between 7 and 11, because we set the thread number equal to 8. When a file has more than 8 fragments, the first 8 fragments are concurrently processed by the multi-threads while the other fragments are waiting to be served. We can conclude from Fig. 1.9(a)-Fig. 1.9(b) that using multi-threading method can improve the throughput compared with the traditional scheme.

1.8 CONCLUSION

It is critical to maintain the confidentiality of files stored in a distributed storage system, even when some storage nodes in the system are compromised by attackers. In recognizing that storage nodes in a distributed system have heterogeneous vulnerabilities, we investigated a secure fragment allocation scheme by incorporating secret sharing and heterogeneous vulnerability to improve security of distributed storage systems.

We addressed the security heterogeneity issue by categorizing storage servers into different server-type groups (or server group for short), each of which represents a level of security vulnerability. With heterogeneous vulnerabilities in place, we developed a fragment allocation scheme called S-FAS to improve security of a heterogeneous distributed system. S-FAS allocates fragments of a file in such a way that even if attackers compromised a number of server groups but fewer than k fragments are disclosed, the file cannot be reconstructed from the compromised fragments.

To evaluate the S-FAS scheme, we built the static and dynamic assurance models in order to quantify the assurance of a heterogeneous distributed storage system processing file fragments. We developed a SAP file allocation algorithm based on the analysis of the assurance model as well as the proposed S-FAS scheme. In order to measure the performance of the S-FAS scheme and the algorithm, we built a prototype in a real-world distributed storage system.

We demonstrated how S-FAS incorporates the vulnerability heterogeneity feature into file fragment allocation for distributed storage systems. Experimental results show that increasing heterogeneity levels can improve file

assurance in a distributed storage system. The experimental results of our prototype implementation offer us inspiration on how to use S-FAS to efficiently improve security and performance in distributed storage systems with heterogeneous vulnerabilities.

Acknowledgments

This research was supported by the U.S. National Science Foundation under Grants CCF-0845257 (CAREER), CNS-0917137 (CSR), CNS-0757778 (CSR), CCF-0742187 (CPA), CNS-0831502 (CyberTrust), CNS-0855251 (CRI), OCI-0753305 (CI-TEAM), DUE-0837341 (CCLI), and DUE-0830831 (SFS), as well as Auburn University under a startup grant, and a gift (No. 2005-04-070) from the Intel Corporation.

REFERENCES

1. S. Jajodia A. Mei, L. V. Mancini. Secure dynamic fragment and replica allocation in large-scale distributed file systems. *IEEE Trans. on Parallel and Distributed Systems*, 14(9):885 – 896, Sept. 2003.
2. J. F. Bouchard D. L. Kewley. Darpa information assurance program dynamic defense experiment summary. *IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans*, 31(4):331 –336, Jul 2001.
3. F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with cfs. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 202–215, New York, NY, USA, 2001. ACM.
4. J. D. Strunk G. R. Ganger H. Kiliccote P. K. Khosla J. J. Wylie, M. W. Bigrigg. Survivable information storage systems. *Computer*, 33(8):61 –68, Aug 2000.
5. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: an architecture for global-scale persistent storage. *SIGPLAN Not.*, 35:190–201, November 2000.
6. S. Lakshmanan, M. Ahamad, and H. Venkateswaran. Responsive security for stored data. *IEEE Trans. on Parallel and Distributed Systems*, 14(9):818 – 828, 2003.
7. H. Mantel. On the composition of secure systems. In *2002 IEEE Symposium on Security and Privacy.*, 2002.

8. M. O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM*, 36:335–348, April 1989.

9. A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

10. B. M. Thuraisingham and J. A. Maurer. Information survivability for evolvable and adaptable real-time command and control systems. *Knowledge and Data Engineering, IEEE Transactions on*, 11(1):228 –238, 1999.

11. W. Yurcik, G. A. Koenig, X. Meng, and J. Greenseid. Cluster security as a unique problem with emergent properties: Issues and techniques. In *5th LCI International Conference on Linux Clusters: The HPC Revolution 2004*, pages 18–20, 2004.

12. G. Zanin, A. Mei, and L. V. Mancini. Towards a secure dynamic allocation of files in large scale distributed file systems. In *HOT-P2P '04: Proceedings of the 2004 International Workshop on Hot Topics in Peer-to-Peer Systems*, pages 102–107, Washington, DC, USA, 2004. IEEE Computer Society.

13. George Copeland, William Alexander, Ellen Boughter, and Tom Keller. Data placement in bubba. *SIGMOD Rec.*, 17(3):99–108, 1988.

14. James F. Kurose and Rahul Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Trans. Comput.*, 38(5):705–717, 1989.

15. Lin-Wen Lee, Peter Scheuermann, and Radek Vingralek. File assignment in parallel i/o systems with minimal variance of service time. *IEEE Trans. Comput.*, 49(2):127–140, 2000.

16. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '91, pages 129–140, London, UK, 1992. Springer-Verlag.

17. M. Pourzandi, D. Gordon, W. Yurcik, and G. A. Koenig. Clusters and security: distributed security for distributed systems. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 1, pages 96 – 104 Vol. 1, May 2005.

18. Peter Scheuermann, Gerhard Weikum, and Peter Zabback. Data partitioning and load balancing in parallel disk systems. *The VLDB Journal*, 7(1):48–66, 1998.

19. G. J. Simmons. How to (really) share a secret. In *CRYPTO '88: Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology*, pages 390–448, London, UK, 1990. Springer-Verlag.

20. M. Tu, P. Li, I-Ling Yen, B. M. Thuraisingham, and L. Khan. Secure data objects replication in data grid. *IEEE Trans. on Dependable and Secure Computing*, 7(1):50 –64, 2010.

21. T. Wu, M. Malkin, and D. Boneh. Building intrusion tolerant applications. In *SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium*, pages 7–7, Berkeley, CA, USA, 1999. USENIX Association.