Distributed Secure Virtual File System Using FUSE

Shin Tezuka[1], Akifumi Inoue[2], Ryuya Uda[2], Kenichi Okada[3]

[1] Graduate School of Science and Technology, Keio University, 3-14-1 Hiyoshi Kohoku-ku, Yokohama-shi Kanagawa-ken 223-8522, Japan,
[2] School of Computer Science, Tokyo University of Technology, 1404-1 Katakuramachi, Hachioji-shi, Tokyo 192-0982, Japan,
[3] Faculty of Science and Technology, Keio University, 3-14-1 Hiyoshi Kohoku-ku, Yokohama-shi Kanagawa-ken 223-8522, Japan,
e-mail: tezuka@mos.ics.keio.ac.jp, akifumi@cs.teu.ac.jp, uda@cc.teu.ac.jp, okada@ics.keio.ac.jp

Abstract In this paper, we describe the implementation and evaluation of the distributed secure file system based on FUSE. A tremendous amount of content is now saved on file servers. File servers and network storage can also easily be installed in small organizations by using operating systems such as Linux. However, the storage capacity of PCs that are generally used is also increasing due to low-priced hard disks. The surplus disk capacity of these PCs is also increasing rapidly especially for work and research. Therefore, we need to use these surplus disks effectively by using distributed preservation of files. Furthermore, administrators may also inadvertently contribute to information leakage and falsification. Therefore, it has recently become necessary to ensure not only user authentication and communication pathways but security within the medium used for storage. Therefore, we implemented and evaluated a system we called JIGFS, which accomplishes secure file management based on a public key for all users. It also distributes and saves file using common PCs, and efficiently uses surplus disk capacity and the communication band. As it is implemented using Filesystem in USErspace (FUSE), users can use it like a conventional file system.

Please use the following format when citing this chapter:

Tezuka, S., Inoue, A., Uda, R., Okada, K., 2008 in IFIP International Federation for Information Processing, Volume 286, Towards Sustainable Society on Ubiquitous Networks, eds. Oya, M., Uda, R., Yasunobu, C., (Boston: Springer), pp. 161–172.

1 Introduction

Most companies and organizations now treat many forms of content as digital data. Also, file servers and network storage in small organizations can easily be built using operating systems such as Linux. However, the storage capacity of general use PCs (not servers) is also increasing due to low-priced hard disk drives. The surplus disk capacity of these PCs is also increasing rapidly especially for work and research. Therefore, we need to use these surplus disks effectively by using distributed preservation of files. However, when using file servers, a person has to manage these. This makes leakage or falsification of files possible if management is negligent. Finding a reliable manager can be difficult for small and medium-sized companies. Moreover, security breaches may also be discovered when servers are installed. Furthermore, administrators may also contribute to leaked and falsified information. Therefore, it has recently become necessary to ensure not only user authentication and communication pathways but the security within the medium used for storage.
As a result, we implemented and evaluated a system that we called the JIGsaw puzzle File System (JIGFS). This system achieves secure file management based on a public key for all users. It also distributes and saves files using common PCs, and uses surplus disk capacity and the communication band more efficiently. Moreover, as it is implemented based on FUSE [1], users can use it like conventional file systems.
The remainder of this paper is as follows: Sections 2 describes related work; Section 3 explains the design and implementation details; Section 4 portrays the evaluation; Section 5 discusses deployment and Section 6 concludes the paper.

2 Related work

In other related work, there have been many systems that distribute and save files to various PCs. However, security has not been a serious consideration, and these systems have been exposed to the risk of leaked information. Increased efficiency in disk capacity was also not taken into consideration as files were duplicated. // For example, Network File Systems (NFSs) [2] do not encrypt the file itself, transmit data, or ensure security between hosts. In contrast, there are systems that authenticate and encrypt transmitted data based on a Public Key Infrastructure (PKI), such as the Self-certifying File System (SFS) [3], the Secure SHell (SSH) File system [4] and the Grid Data farm (Gfarm) [5] that use grid technology and the Grid Security Infrastructure and Self-certifying File System (GSI-SFS) [6]. However in these systems, the performance is considered as important to treat large files, and do not have files encrypted within the media used to store them in their final locations. Therefore, they have not considered files in storage media being physically stolen or leaked by administrators who have authority to access servers.
There is also a system implemented as a file server with careful attention to security by our precedence work [7]. However, this system is not user friendly, because the

user needs to use client software like FTP. Thus, the new type of secure distributed file system is needed.

3 Distributed secure virtual file system

This section explains design and implementation of distributed secure virtual file system. How files are acquired and preserved with this system is introduced, and the criteria by which keys are managed and distribution places are selected to preserve them are described after that.

3.1 Outline of our system (JIGFS)

This section provides an outline of our system called JIGsaw puzzle File System (JIGFS), which allows effective sharing of communication bands and disk space, and creates a huge virtual file system that enables large-scale file distribution and mass storage. It uses two or more local area networks (LANs) and cooperates with a File System Gateway (FSG) as outlined in Fig.1. It differs from a general file server or network storage, and a file that is saved by this system is encrypted on a user's PC and divided into multiple data files with redundancy on an FSG, which are then saved in user PCs belonging to other LANs. Any falsification can be discovered through hash values. Thus, this system has a virtual RAID structure and a high level of security. This approach also enables disk space and communication bands to be efficiently shared.
All data regarding files and user information is saved in a database on the FSG. To

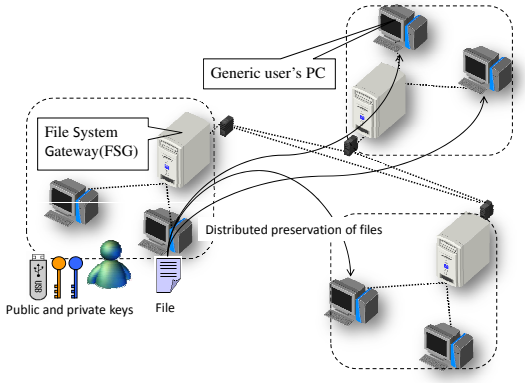

Fig. 1 Outline of our system

protect against failures, a database can be backed up by saving it in other networks as can be done for conventional files. The private and public keys needed to access files in this system are stored in small removable devices such as USB memories. Such devices are easy for users to carry with them and allow them to access the system from any PC, not only the PC they usually use. When a user mounts JIGFS, the FSG to which he belongs authenticates him with his public key and private key using challenge and response. And also, the user's public key is stored in the database with a certification by a certification agent (CA).

3.2 Saving files through FSG

When user application writes a file with this system through the following procedure and there are sequence diagrams of this in Fig.2.
First, a file is passed to JIGFS through FUSE like step 1 in the figure. And it is encrypted with a random key on the user's PC with a common key cryptosystem like step 2. This file is then sent to the FSG, along with file properties such as the file name, permission, the hash value of the file by Secure Hash Algorithm 1 (SHA-1, 160bits), and the common key. These properties and the common key are encrypted by the public key cryptosystem (See sect.3.4 for details). The FSG receiving the file caches it own disk with Least Recently Used (LRU) like step 3 in the figure. After that, FSG divides this into a fragment of two or more information units called shares with redundancy by using an Information Dispersal Algorithm (IDA) [8] described on step 4. And sends this to another FSG that has sufficient disk capacity and the available communication bands like step 5. Furthermore, an FSG that receives the distributed shares saves the data in the most compatible PC with respect to considerations such as boot-time characteristics. Finally, the FSG stores the file properties and information as to where the shares are saved to its own database. Moreover, when a saved file is shared with a group, the properties are saved to a database encrypted with the group's public key.

3.3 Retrieving files through FSG

A file is retrieved through a procedure that is the reverse of the save operation. The user's PC obtains the properties of the saved file from the FSG and decrypts these with the user's private key. If the FSG has a file as a cache, the FSG sends the file to the user's PC. When the file has been distributed, information such as the file name, permission, and the location of the shares for the file is sent to the FSG. The FSG sends search requests for the shares to other FSGs, which search for and collect the shares from PCs belonging to their networks that have parts of the target file. When the number of collected shares exceeds a threshold, the FSG restores the file from divided shares and sends it to the user's PC. After receiving the file, the user's PC

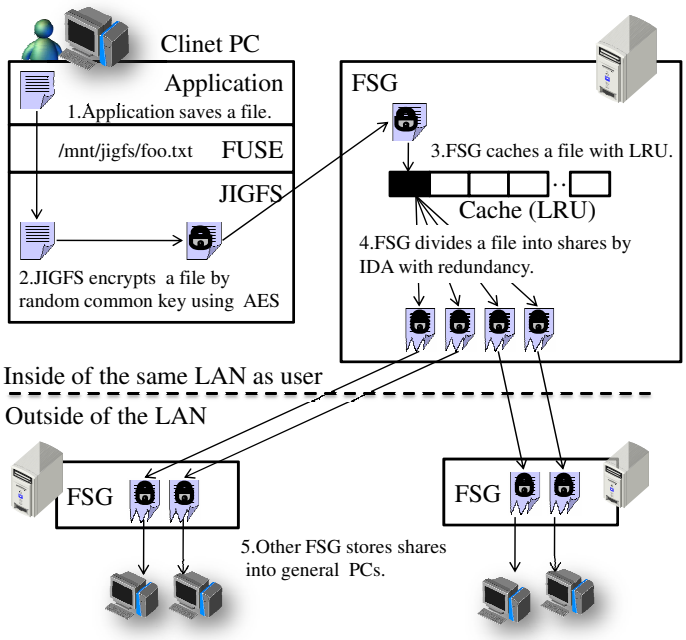

Fig. 2 Saving file through FSG

then decrypts the common key from the file properties with the user's private key. Finally, it decrypts the file with the common key and the user can access the file. The information integrity of the file can be confirmed by using a hash value for falsification requested when the file was saved. In addition, even if all the shares cannot be retrieved, the file can still be restored with less than the threshold number of shares because of redundancy in saving the file.

3.4 Common key and file properties

The random common key used to encrypt a file and the file properties in this system are encrypted with Rivest Shamir Adleman (RSA, key length: 1024bit) [9] on a user's PC and sent to an FSG. Although RSA code reinforcement is high, this process requires a long time. Thus, file encryption uses AES (key length: 128-bits) [10] with a random common key. The greatest concern is how the key is managed. After the body of the file has been encrypted using AES and the file properties and the key are encrypted in RSA, they are stored in a database on an FSG as outlined in Fig.3. This ensures the information is securely managed and the only the user's private key is needed to retrieve a file.

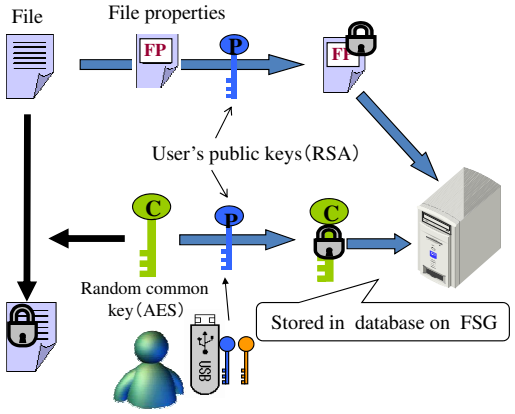

Fig. 3 Management of common key and file properties

3.5 Valuation function

The networks where shares are saved are determined by Eq.1. The purpose of this valuation function is to increase the recovery rate of shares and to effectively use the surplus disk area of all PCs, and the communication band between networks. A value is calculated to evaluate other FSGs from the FSG to which the user belongs. V means the surplus disk area, T means the throughput between networks and R means reliability. Moreover, w is a weighting factor that is determined from Eq.2 in a range from 0 to 1.0 according to the size and the reference frequency of a file. However, when the file size is larger than 4 Gbytes, consume a great deal of disk capacity, it is uniformly referred to as $w = 1.0$.

$$E = \{wV + (1-w)T\} \times R \tag{1}$$

$$w = \frac{\log(F_{size})}{10} \quad (0 \leq w \leq 1.0) \tag{2}$$

1. Boot-time characteristics
   The boot-time characteristics express by a bit series in what kind of time zone the user and PC have booted and joined the network. The period to express these characteristics is one day, and the grain size is 5-min intervals. One means a boot state and zero means an idle state in these characteristics. Therefore, 288 bits/day are used for expressing the boot-time characteristics per PC or user. This is saved for five weeks, and when all evaluations are calculated, decision by majority is used for all bits. For example, when the user was joined between 0:00-0:05 three or more days on Monday, the first bit of the boot-time characteristics is set to one.

2. Disk-capacity V
   V indicates how much disk area can be used, when a share is stored in a network. The user and the boot-time characteristics of his or her PC and surplus disk capacity are taken into consideration when calculating this value. For example, taking the user of boot-time characteristic S into account saves a file where there is a network A, B, or C used as a place to preserve it. The number of PCs belonging to a network is set to n, and the primary value of evaluating each network is calculated from Eq.3. Furthermore, the value to evaluate the disk capacity of Network A V_A is calculated by doing it relatively from Eq.4. In addition, $\mid S \mid$ and $\mid D \mid$ are the sum totals of the number of ones among the bit series that have boot-time characteristics, and $\mid S \wedge D_i \mid$ is the sum total of the bit set and one among those bitwise AND's.

$$V'_A = \sum_{i=1}^{n} \frac{\mid S \wedge D_i \mid}{\mid S \mid} C_{D_i} \tag{3}$$

$$V_A = \frac{V'_A}{V'_A + V'_B + V'_C} \tag{4}$$

3. Network throughput T
   This system measures the throughput of a network when some FSG downloads a share from a partner's FSG each time, and accumulates it. This value is saved for one month for every day of the week. For example, when there is a network A, B, and C, the throughput when FSG A downloads a share from FSG B is set to TP_{AB}; the throughput when downloading from FSG C is set to TP_{AC}. In this case, the throughput-evaluation value T_{AB} of FSG B assessed from FSG A is Eq.5.

$$T_{AB} = \frac{TP_{AB}}{TP_{AB} + TP_{AC}} \tag{5}$$

4. Reliability R
   R, which indicates the reliability of some network is calculated from Eq.8 This is used when calculating the evaluation value, E, of each network. The hash value of the saved share is periodically verified between FSGs. Tr calculated from Eq.6 is the rate of the number of times Hn that the partner's FSG answered correctly to the number of times Hc verified the hash value of the share. Moreover, Ma, which was calculated from Eq.7, expresses the rate of the number of PCs specified by m that boot in parallel with the user's PC.

$$Tr = \frac{Hc}{Hn} \tag{6}$$

$$Ma = \frac{1}{m} \times \sum_{i=1}^{m} \frac{\mid S \wedge D_i \mid}{\mid S \mid} \tag{7}$$

$$R = Ma \times Tr \tag{8}$$

5. Number of shares, and threshold
   Threshold k is determined by specifying the section with the original file size to make the share size after division by IDA regular. For example, when the original file size is 1 MB, the threshold is determined to be $k = 10$ so that share size can be set to 100 KB. Therefore, even if someone collects shares of comparable size, the possibility of restoring the original file is low. Since threshold k was previously determined, the redundancy of a file is determined by n, which is the number of divisions of a file. n is calculated from an Eq.9 using Ma. However, n is a natural number.
   The number of shares stored in each network is determined by the relative rate of the evaluation value, E. For example, n is the total number of shares, E_A is the evaluation value for Network A, and E_{sum} is the total evaluation value for all the networks. In this case, n_A, which indicates the number of shares distributed to Network A is calculated from Eq.10. However, n_A is a natural number.

$$n = [\frac{k}{Ma}] \tag{9}$$

$$n_A = [n \times \frac{E_A}{E_{sum}}] \tag{10}$$

4 Evaluation

4.1 Evaluation of file writing and reading

We evaluated file writing and reading using FUSE on this system. LANs A, B, and C were connected to provide the experimental environment. Three sets of PCs were connected to each LAN. One set of PCs belonging to LAN-A wrote and read files. Some of the PCs belonging to LANs-B and -C were at the storage location of the final shares. The specifications of the PCs used for evaluation are OS: Linux 2.4.31, CPU: Celeron 2.8GHz, Memory: 512MB, HDD: 120 GB, 5400 rpm, Cache: 8 MB (including FSG). All the communication bands connected to all PCs are 1 GBASE-T. The number of shares at the time they were divided by IDA was set to $n = 5$, and the threshold was set to $k = 3$.

Fig.4 plots the times for writing and reading files of various sizes. These times include the distribution of shares at the time of writing, and the collection of shares at the time of reading. We can see that this system is practical for files that contain a few megabytes, but by the time they reach 50 MB, it is no longer high speed, since about 180 sec is required. Based on these results, evaluation using the cache method on FSG is considered to be much faster, as shown in Fig.5. Even files of 50 MB take little more than 10 sec, and this is much faster than where distributed preservation of shares is included. Therefore, it is important to balance the risk of basing distributed

preservation of shares and effective use of resources with the rapidity by the cache method.

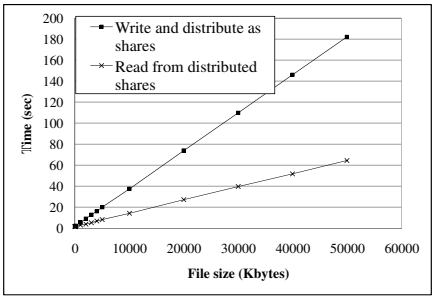

Fig. 4 Processing times for writing and reading files (Distribution of shares is included)

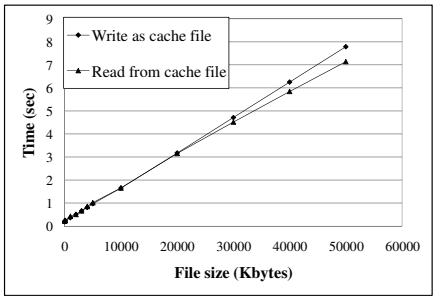

Fig. 5 Processing times for writing and reading files (Using cache on FSG)

4.2 Evaluation by simulation

Large-scale experiments are difficult to assess this system using two or more networks and numerous PCs. Therefore, the acquisition rate for successful file access and the cache hit ratio were simulated with a valuation function. The simulation environment is summarized in Table1 and LANs A-E were connected. Each PC belonging to the networks had 10 GB of surplus disk area, and the simulation took two weeks. Each users saved ten files per day, and restored them. We based the size of files to save on J. R. Douceur et al. [11] who investigated their sizes in various file systems. We used Sun Microsystems Inc.'s statistics for the re-reference frequency of files to restore [12]. Weighting factor w of a valuation function determines the initial value for the size of a file. After this, 0.1 is added every time a file is referred to.

We conducted the simulations based on the boot-time characteristics of real PCs. The boot-time characteristics were obtained from boot logs collected from five locations, such as laboratories in university, for two weeks over a six-month period. The transition in the number PCs booted during the two weeks is plotted in Fig.6. Since it was Sunday on the 1st, the 1st, 7th, 8th, and 14th occurred on weekends. Therefore, fewer PCs were booted than on weekdays. Because there were many users, especially in laboratories, who left their PCs without logging out or turning off the power, the number of PCs that booted around mid night was not completely zero.

Fig.7 shows the results for the acquisition rate. After simulation began, the acquisition rate to hit caches reached 100% within two days. After this, the hit ratio for caches fell gradually and the number of shares distributed in the PCs of other

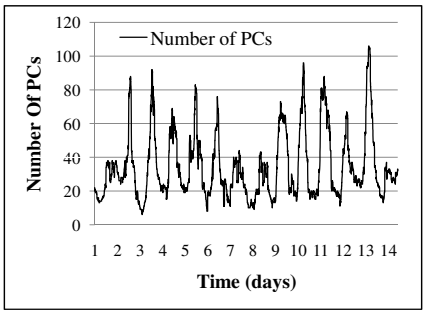

Fig. 6 Transition in number of booted PCs

Table 1 Simulation environment

Network	Number of PCs	Bandwidth(Kbps)	Cache(MB)
A	10	1000000	100
B	20	100000	200
C	30	10000	300
D	40	1000	400
E	50	100	500

networks to access files increased. Because the number of shares currently collected was less than the threshold to restore the original files by using IDA, the number of cases where files could not be accessed increased. The final acquisition rate reached 90% or more, where the cache hit ratio was about 50%. The relative capacity of shares stored in four networks (A-E) is plotted in Fig.8. We can see that although the distributed preservation of shares was concentrated on network A where there is a large communication band in the early stage, this is gradually rearranged in networks with a great deal of surplus disk capacity.

5 Discussions

In this system, the division of files by using IDA is slower than the throughput of encryption or that of the network. However, encrypted files are held once during a fixed term as a cache in FSG. Therefore, from the user's viewpoint, when the preservation of a file causes an encrypted file to be sent FSG, the process is completed. Moreover, since it is implemented as a virtual file system through FUSE, control returns immediately when the file is closed. Therefore, the response time that a user experiences is only the time to encrypt and send a file to an FSG. From the results in Fig.4, writing is practical if it is completed in a little more than 10 sec for a file of tens of megabytes, even if this is a few seconds less for a file of a few megabytes. The processing times differ according to whether it is necessary to read files, store

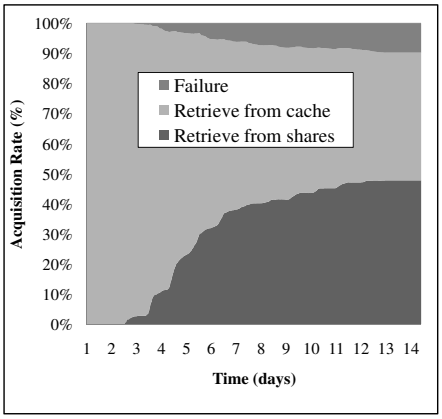

Fig. 7 Transition in acquisition rate

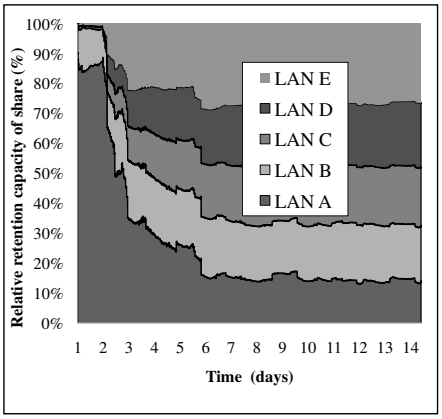

Fig. 8 Relative capacity of stored shares

them in an FSG as a cache, or collect shares, as shown in Fig.4 and Fig.5. When it is necessary to collect shares, it takes about 60 sec to open a file when the file size is 50 MB, about 12 sec to open 10 MB, and about 5 sec to open 1 MB where shares are distributed and saved in the network. When 50-MB files are stored in an FSG as a cache, it only takes about 12 sec to send them to a client PC and decrypt them at high speed. Thus, this system effectively uses resources by using distributed preservation of shares, and risk is reduced when obstacles occur. It can simultaneously be employed in practice by efficiently using the cache function of an FSG.

This system assumes an environment where all PCs are connected and can leave the network at any time; it arranges shares by taking boot-time characteristics into consideration. From the results of simulation, presented in Fig.7, which used the boot-time characteristic of real PCs, files were able to be accessed with a probability of 90% or more. Files in the simulation were rearranged periodically by recalculating the evaluation value of saved files. Fig.8 shows the relative capacity of the total number of shares stored in each network. Network A's throughput is particularly important immediately after this system has started. However, files that have not been referred to for a long time are rearranged to network E, which has surplus disk capacity. According to this structure, resources can be efficiently used from both respects of disk capacity and communication band.

6 Conclusion and Future works

This system manages files based on a public key for all users, encrypts the files, and provides them with distributed preservation. Thereby, it can reduce costs by not having to employ experienced administrators under exclusive contracts who understand advanced technologies. Moreover, as this was implemented using FUSE, users can use it together with conventional file systems. Therefore, users do not have to be

conscious of the existence of this system. Moreover, we demonstrated that it could efficiently be used in a general environment, such as companies, from the results of a simulation based on a log showing the boot time characteristics of a PC we actually used. Users can also use it practically on a daily basis with small-scale files.
However, to treat large-scale files that exceed 1 GB at high speed, it is necessary to improve performance, such as replacing the method of division with an IDA, the method of caching, and take file properties into consideration.
On the other hand, depending on timing, a part of files may be unable to be retrieved temporarily. Therefore, we consider that FSG should starts PCs which has a share using Wake-On-Lan as future work. Thereby, a user can always acquire a file, and it is not necessary to boot useless PCs. Moreover, although the system is currently installed as a virtual file system for Linux using FUSE, we want to find whether an equivalent function can also be achieved in the Windows environment.

Acknowledgements This work is supported in part by a special grant from SECOM Science and Technology Foundation.

References

1. Filesystem in Userspace.: http://fuse.sourceforge.net
2. Inc. Sun Microsystems.: NFS, Network File System Protocol Speciation. RFC1094 (1989).
3. D. Mazieres.: Self-certifying File System. Ph.D. thesis, Massachusetts Institute of Technology (2000).
4. SSH Filesystem.: http://fuse.sourceforge.net/sshfs.html
5. O. Takebe, N. Soda, and S. Sekiguchi.: Gfarm v2: Design and Implementation of Global Virtual File System. IPSJ SIG Notes, 2004-HPC-099, pp.145-150 (2004).
6. S. Takeda, S. Date, and S. Shimojo.: A User-oriented Secure Filesystem on the Grid. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2003) in Tokyo, Japan (2003).
7. S. Tezuka, R. Uda, A. Inoue, and Y. Matsushita.: A Secure Virtual File Server with P2P Connection to a Large-Scale Network. The IASTED International Conference on Networks and Communication Systems, No.527-138 (2006).
8. M. O. Rabin.: Efficient dispersal of information for security, load balancing, and fault tolerance. J.ACM, Vol. 36, pp.335–348 (1989).
9. R. L. Rivest, A. Shamir, L. M. Adleman.: A Method for Obtaining Digital Signatures and Public-key Cryptosystems. Communications of the ACM, 21, pp.120-126 (1978).
10. J. Daemen and V. Rijmen.: AES proposal: Rijndael, AES algorithm submission. http://nist.gov/aes (1999).
11. J. R. Douceur and W. J. Bolosky.: A large-scale study of file-system contents. ACM SIGMETRICS Performance Evaluation Review, vol.27, pp.59-70 (1999).
12. Inc. Sun Microsystems.: Information Lifecycle Management Technical Overview. http://government.hp.com/