Performance Evaluation of a Disaster Recovery System and Practical Network Applications in Cloud Computing Envionments

Yoichiro Ueno, Noriharu Miyaho, Shuichi Suzuki
School of Information Environment,
Tokyo Denki University,
Muzai Gakuendai, Inzai-shi, Chiba, 270-1382 Japan
e-mail: ueno@sie.dendai.ac.jp,
miyaho@sie.dendai.ac.jp, ssuzuki@sie.dendai.ac.jp

Kazuo Ichihara
Net & Logic. Co.
Daizawa, Setagaya-ward, Tokyo, 155-0032 Japan
e-mail: Ichihara@nogic.net

***Abstract* — This paper presents evaluation results for a high security disaster recovery system using distribution and rake technology. In an experimental evaluation, the encryption and spatial scrambling performance and the average response time have been estimated in terms of the data file size. Discussion is also provided on an effective shuffling algorithm to determine the dispersed location sites. Finally, this paper describes a prototype system configuration for several practical network applications, including the hybrid utilization of cloud computing facilities and environments which are already commercialized.**

Keywords-disaster recovery; backup; metadata; distributed processing; cloud computing; strong cipher; secure video streaming

I. INTRODUCTION

Innovative network technology to guarantee, as far as possible, the security of users' or institutes' massive files of important data from any risks such as an unexpected natural disaster, a cyber-terrorism attack, etc., is becoming more indispensable day by day. To meet this need, technology is required that can be used to realize a system that has affordable maintenance and operation costs and provides high security.

For this application, Data Grid technology is expected to provide an effective and economical backup system by making use of a very large number of PCs whose resources are not fully utilized. In particular, Data Grid technology using a distributed file data backup mechanism will be utilized by government and municipal offices, hospitals, insurance companies, etc., to guard against the occurrence of unexpected disasters such as earthquakes, large fires and storms.

However, these methods involve high operation costs, and there are many technical issues to be solved, in particular relating to security and prompt restoration in the event of disasters occurring in multiple geographical locations.

In addition, there is a network infrastructure, which can be used to distribute and back up a great number of data files, and a large number of remote office personal computers, smart phones, or cellular phones, can be readily utilized for this purpose.

In addition to these factors, many people involved in industry and commerce are interested in making use of public or private cloud computing facilities/environments, provided by carriers or computer vendors, to provide temporary storage of and real-time access to their multimedia files, as a means of achieving security and low maintenance and operation costs. However, the guaranteed security level provided by cloud computer services has not yet been established.

In this paper we propose an innovative file backup concept which makes use of an effective ultra-widely distributed data transfer mechanism and a high-speed encryption technology, based on the assumption that we can use a small portion of the storage capacity of a large number of PCs and cellular phones that are in use in daily life, to efficiently realize safe data backup at an affordable maintenance and operation cost [1][2].

Figure 1 shows a comparison of the proposed method with the conventional method in terms of network utilization. The principal differences in the proposed system are as follows. (1) It does not require the use of expensive leased lines. (2) It only utilizes otherwise unused network resources, such as unused network bandwidth and unused storage

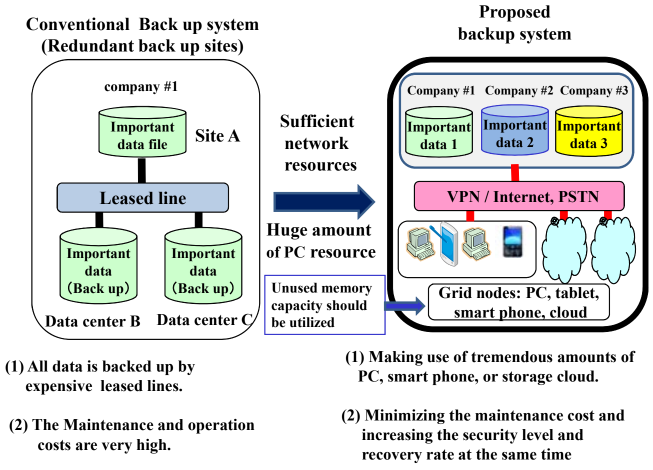
Conventional Back up system
(Redundant back up sites)
company #1
Important data file
Site A
Leased line
Important data (Back up)
Important data (Back up)
Data center B
Data center C
Sufficient network resources
Huge amount of PC resource
Unused memory capacity should be utilized
Proposed backup system
Company #1
Company #2
Company #3
Important data 1
Important data 2
Important data 3
VPN / Internet, PSTN
Grid nodes: PC, tablet, smart phone, cloud
(1) All data is backed up by expensive leased lines.
(2) The Maintenance and operation costs are very high.
(1) Making use of tremendous amounts of PC, smart phone, or storage cloud.
(2) Minimizing the maintenance cost and increasing the security level and recovery rate at the same time

Figure 1. Comparison of the proposed system with a conventional data backup system

capacity in PCs, smart phones and cellular phones, etc. (3) It can utilize cloud computing facilities/environments as one of the remote Grid nodes to obtain a requested amount of storage and a specific security level in accordance with the customer's requirements. (4) It can cipher a number of important data files at the same time using spatial scrambling and random dispatching technology. (5) As the number of user companies increases, the security against being deciphered illegally increases accordingly. (6) The maintenance cost can be drastically reduced. In addition, since it uses a stream cipher, the speed of encryption of data is increased, so it can also be applied to secure streaming for video transmission services.

In general, encryption can use two types of technology, that is, block cipher technology or stream cipher technology. In the case of block cipher technology, the data is divided into successive blocks and each block is processed separately for point-to-point systems; as a result, the encryption speed is quite slow. As the data volume increases, the required processor and memory cost increases in an exponential manner.

On the other hand, in case of the stream cipher, the input data is simply operated on bit-by-bit, using a simple arithmetic operation. Therefore, high-speed processing becomes feasible. These are the fundamental differences between the two cipher technologies [3].

When an ultra-widely distributed file data transfer technology, a file fragmentation technology and an encryption technology are used together, then quite different characteristics arise from a point of cipher strength. It is possible to combine the use of technologies, specifically, the spatial scrambling of all data files, the random fragmentation of the data files, and the corresponding encryption and replication of each file fragment using a stream cipher. The corresponding history data, including the encryption key code sequence, which we call "encryption metadata", are used to recover the original data. This mechanism is equivalent to realizing a strong cipher code, comparable to any conventionally developed cipher code, by appropriately assigning affordable network resources [4].

By making use of the proposed combined technology, illegal interception and decoding of the data by a third party becomes almost impossible and an extremely safe data backup system can be realized at reasonable cost. The proposed technology can also increase both the cipher code strength and the data recovery rate.

To realize the proposed disaster recovery mechanism, the following three principal network components are required: (1) a secure data center, (2) several secure supervisory servers, and (3) a number of client nodes such as PCs or cellular phones. We have previously clarified the relationships between data file capacity, the number of file fragments and the number of replications for the case of disaster recovery [5][6].

In this paper, we briefly describe the related work in Section II, and discuss the basic configuration of the system architecture in Section III, and the performance evaluation in Section IV. The practical experimental system is discussed in Section V. Finally, we provide our conclusions from these studies in Section VI.

II. RELATED WORK

Conventionally, a file data backup system [7] has been realized by duplication of a data center, access lines, etc. However, considering that an earthquake may cause fiber cable failures over a wide area and shut down several communication links, this approach is not fully satisfactory [8].

To take account of this, we have already proposed a disaster recovery concept which can use to realize a reliable file backup system by making use of safe and fast encryption mechanisms, a network distribution mechanism, and a technique for the effective use of secure meta-data containing the history of encryption key code sequences [9][10][11][12]. This technical approach has not yet been implemented elsewhere. Other related studies have included the concept of a distributed file backup system [13][14]. However, in these studies, neither a precise performance evaluation nor practical network service systems are clearly described. In addition, the technological aspects concerning the effective distribution of the fragmented data to establish the required security level are not clearly discussed for conventional systems.

On the other hand, there has been only a little published research on personal disaster recovery systems. A personal disaster recovery system should take account of the cost and the guarantee of security of personal data. So, a personal disaster recovery system may use some kind of cheap storage service for data backup. In recent years, it has become possible to implement data backup using free online storage. However, such low cost storage services are not secure enough for personal information. A persistent file server that should improve security of such storage service has been proposed [15]. This persistent file server system introduced encryption, fragmentation, replication, and scattering. Our proposed system adds more security with spatial scrambling, a second encryption, and an effective shuffling algorithm.

III. BASIC CONFIGURATION OF THE SYSTEM ARCHITECTURE AND ITS VARIATIONS

This section discusses the basic configuration of the high security distribution and rake technology (HS-DRT).

A. *Basic Configuration of the HS-DRT System*

The HS-DRT file backup mechanism has three principal components as shown in Figure 2. The main functions of the proposed network components are Data Center, Supervisory Server and various client nodes, and these can be specified as follows.

The client nodes (at the bottom of Fig. 2) are PCs, Smart Phones, Network Attached Storage (NAS) devices, and Storage Services. They are connected to a Supervisory Server in addition to the Data Center via a secure network.

The Supervisory Server (on the right in Fig. 2) acquires the history data, which includes the encryption key code sequence (metadata) from the Data Center (on the left in Fig. 2) via a network.

The basic procedure in the proposed network system is as follows.

1) Backup sequence

When the Data Center receives the data to be backed up, it encrypts it, scrambles it, and divides it into fragments, and thereafter replicates the data to the extent necessary to satisfy the required recovery rate according to the pre-determined service level. The Data Center encrypts the fragments again in the second stage and distributes them to the client nodes in a random order. At the sane time, the Data Center sends the metadata used for deciphering the series of fragments to the Supervisory Server. The metadata comprises encryption keys (for both the first and second stages), and several items of information related to fragmentation, replication, and distribution.

2) Recovery sequence

When a disaster occurs, or at other times, the Supervisory Server initiates the recovery sequence. The Supervisory Server collects the encrypted fragments from various appropriate clients in a manner similar to a rake reception procedure. When the Supervisory Server has collected a sufficient number of encrypted fragments, which is not necessarily all encrypted fragments, they are decrypted, merged, and descrambled in the reverse order of that performed at the Data Center and the decryption is then complete. Though these processes, the Supervisory Server can recover the original data that has been backed-up.

B. Secuirty Level of HS-DRT

The Security level of the HS-DRT does not only depend on the cryptographic technology but also on the combined method of specifying the three factors, that is, spatial scrambling, fragmentation/replication, and the shuffling algorithm.

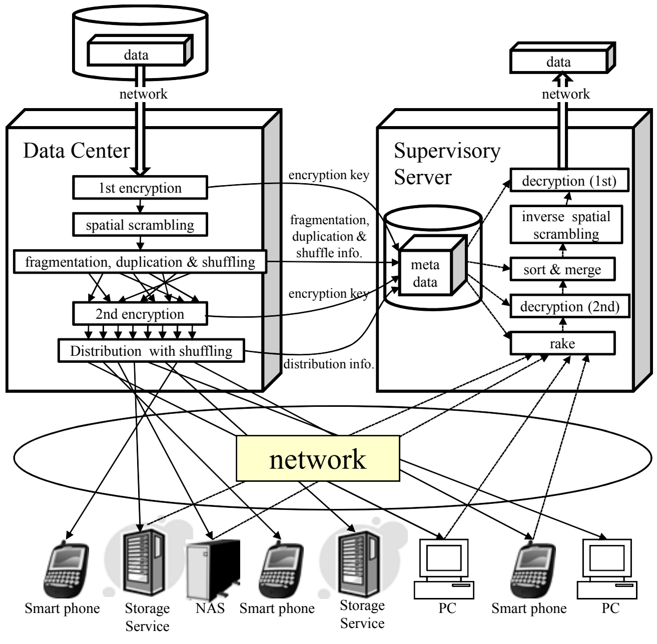

Figure 2. The basic configuration of HS-DRT system

Because of these three factors, nobody is able to decrypt without collecting all relevant fragments, selecting a unique set of fragments, and sorting the fragments into the correct order. Even if some fragments are intercepted, nobody is able to decrypt parts of the original data from such fragments.

1) Spatial scrambling

The spatial scrambling procedure can be realized by executing the simple algorithm illustrated as follows. Using a C-style description, we can write:

```
for(i=1;i<imax;i++){buf[i]=buf[i]+buf[i-1];}
buf[0]=buf[0]+buf[imax-1];
```

The array buf[] is the target data to be scrambled and imax is the size of the buf array. This computation process should be repeated several times. It is strongly recommended that this process be repeated at least six times. To de-scramble, it is only necessary to perform the same operations in the reverse order. By introducing the above mentioned spatial scrambling technology, it is almost impossible for a third party to decipher the data by comparing and combining the encrypted fragments, since uniform distribution of information can be achieved by this spatial scrambling.

2) Fragmentation/replication

One of the innovative ideas of HS-DRT is that the combination of fragmentation and distribution can be achieved in an appropriately shuffled order. Even if a cracker captured all raw packets between the data center and the client nodes, it would be extremely difficult to assemble all the packets in the correct order, because it would be necessary to try about (no. of fragments)! possibilities.

Furthermore, the proposed backup mechanism replicates each fragment and encrypts each copy of the fragment with a different encryption key. Even if a pair of encrypted fragments are the copy of the pre-encrypted fragment with each other, their bit patterns are completely different from each other owing to the different encryption key. Therefore, it is impossible to identify the encrypted fragment with the other. Crackers would require innumerable attempts to decipher the data.

3) Shuffling

HS-DRT mainly uses a shuffling method with pseudo-random number generators for the distribution to the client nodes. When we distribute the fragments of the encrypted data to widely dispersed client nodes, we can send them in a shuffled order, since we predetermine the destination client nodes from the shuffled table in advance. If the shuffled table has a uniform distribution, the table itself is hard for a third part to guess. But, if the shuffled tables are biased, there may exist weak points in the corresponding recovery system.

The result of the significance level, when we divide the data into 100 fragments, is as follows. The significance level for the Mersenne Twister and "Fisher-Yates shuffle" [16][17] with 1 round was 0.4617, and with 3 rounds was 0.8416.

HS-DRT adopts the Fisher-Yates shuffle with 3 rounds as the shuffling method and Mersenne twister as the pseudo-random number generator.

IV. PERFORMANCE EVALUATION

In this section we discuss the encryption performance and the spatial scrambling performance of the core module.

A. *Fundamental data*

We evaluated the performance of three systems. Figure 3 shows the test system, which consisted of two PCs, four Network Attached Storage devices, and three 1000base-T Ethernet networks. Table I shows the test environment adopted in each PC.

We examined the fundamental performance as follows.

By using Mersenne twister (mt19937ar), it was possible to generate a pseudo-random number in 7.35 nsec on PC1. By using the Fisher-Yates shuffle algorithm with 3 rounds and Mersenne twister, it was possible to generate 128 entries for a uniform distribution table in 18.0 μsec on PC1.

By using the 1000base-T (MTU=1522) network and a TCP stream, it was possible to transfer data from PC2 to PC1 at 112 MB/sec. In this paper, we define "MB (MegaByte)" as $(1024)^2$ bytes and "GB (GigaByte)" as $(1024)^3$ bytes, except for HDD capacity. Due to hardware limitations, we did not examine the performance using the jumbo frame size. By using the 1000base-T and FTP protocol, we found that we could achieve 33.9 MB/sec from PC1 to NAS for writing. By using 1000base-T, where the MTU size was 7422, we could achieve 42.6 MB/sec from PC1 to NAS for writing.

B. *Highest performance*

The HS-DRT encryption core module consists of three threads and four buffers. To examine the highest performance, the three threads, that is, the receiver, encryption/scrambling, and sender processes, were as follows. The receiver thread read the dummy data from "/dev/zero". This operation means that the receiver thread reads all zero dummy data without reading from the HDD or the network. The encryption/scrambling thread encrypts and scrambles the data. Then this thread divides the data into 128 fragments, and encrypts them in the shuffled order. The sender thread writes the 128 fragments in the shuffled order to "/dev/null". This operation means that the sender thread abandons all encrypted fragments.

Table II shows the performance of the HS-DRT encryption core module. The size of the buffers was 1GB each. "Time" means the actual elapsed time between the starting time of the first receiver thread and the finishing time of the final sender thread. The HS-DRT encryption core module was able to achieve 150MB/sec or more. This performance was better than that of the 1000base-T network interface of 119MB/sec.

C. *Practical performance*

The results obtained by examining the practical performance of the receiver/sender thread processes are as follows. The receiver thread reads the dummy data from the TCP stream that was transmitted from PC2. The sender

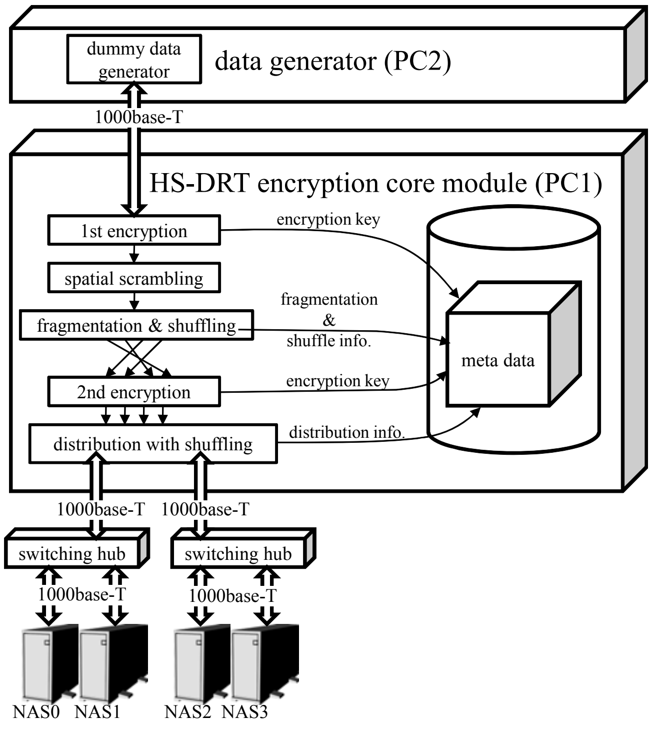

Figure 3. The performance evaluation set-up for the HS-DRT module

Table I. System environment

	PC1 Encryption core module	PC2 Data generator
CPU	Core2 Quad Q6600 2.40GHz	
Memor y	8GB (DDR2-800)	4GB (DDR2-800)
HDD	RAID 0(striping) SATA 500GB 7200rpm×4	SATA 250GB 7200rpm
OS	Fedora 12 x86_64	Fedora 10 i686
Kernel	2.6.31.5-127.fc12.x86_64	2.6.27.5-117.fc10.i686.PAE
gcc	gcc(GCC) 4.4.2 20091027 (Red Hat 4.4..2-7)	gcc (GCC) 4.3.2 20081105 (Red Hat 4.3.2-7)
libc	glibc-2.11-2.x86_64	glibc-2.9-2.i686

Table II. Highest performance of HS-DRT encryption core module

Processed Data size [GB]	Time [sec]	Performance [MB/sec]
1	6.76	151
2	12.8	160
4	25.0	164
8	49.3	166
12	73.9	166
16	97.9	167
24	146	168
32	195	168
48	292	168
64	390	168

thread writes 128 fragments in a shuffled order using a built-in FTP client to four NAS devices. Since the sender thread creates new four threads, the HS-DRT encryption core module can write the encrypted fragments to the NAS devices in parallel.

Although most access lines to the Internet are slower than 100base-T, we used 1000base-T networks in the practical performance evaluations. There are two reasons. First, we hope to use gigabit access lines to the Internet in the near future. The second reason is the system requirement that the backup process should be completed immediately and should not be frustrated by a slow access line. So, the practical HS-DRT system should consist of two stages. In the first stage, the system processes are the 1st encryption, spatial scrambling, and fragmentation, carried out as fast as possible with local area high speed networks. After the first stage, the computer which has been backed up would resume its normal task. Then, in the second stage, the system processes are replication, shuffling, and distribution according to the speed of the access line. In our practical performance evaluation, we focused on the first stage performance.

Table III shows the practical performance of the HS-DRT encryption core module. “Sending Time” means the actual elapsed time for the data transfer from PC2. “Processing Time” means the actual elapsed time between the starting time of the first receiver thread and the finishing time of the final sender thread at PC1. The size of the buffers was 1GB each. “Latency” means the difference between “Sending Time” and “Processing Time”.

The Performance of PC1 changes according to the Processed Data size, and depends on the Latency. As the Latency consists of the final encrypting/scrambling time and the final sending time, it retains a constant value. Figure 4 shows the performance comparison of throughput on PC1. In Fig. 4, the X-axis shows the processed data size in Gigabytes and the Y-axis shows the performance in billions of bits per second. The upper line indicates the throughput of the encryption core module on PC1 under conditions for the highest performance and the lower line shows the same throughput under the practical performance conditions.

The performance degrades at smaller sizes of processed data because of the latency of processing and sending one GB of data. However, this latency becomes negligible at larger processed data sizes. Under the practical performance conditions, the throughput of the encryption core module on PC1 becomes saturated at one billion bits per second. The difference between this one billion bits per second and the total throughput on PC1 may be due to the overhead of Ethernet frame handling.

Figure 5 shows the CPU usage of PC1 and the threads execution timing chart when the HS-DRT encryption core module executes such operations as receiving, processing, and sending 4GB data. In the CPU usage graph in the upper half of Fig. 5, the X-axis indicates the elapsed time and Y-axis indicates the percentage CPU usage. As PC1 has a Quad-Core CPU, 100% of CPU usage means four or more processes/threads running in parallel, and 25% of CPU usage means only one process/thread running. The “IDLE” area shows the percentage of time that the CPU was idle. The “SOFTIRQ” area shows the percentage of time spent by the CPU in handling Soft-IRQs. The “SYS” area shows the percentage of CPU utilization at the kernel level. The “USER” area shows the percentage of CPU utilization at the user level. The total graph area, except for the “IDLE” section, indicates the total CPU workload. Any other parameters, such as the “NICE” value, are omitted from this graph as these values are negligibly small in this performance evaluation.

The “USER” area is the workload of the HS-DRT encryption core module, and the “SYS” + “SOFTIRQ” areas are the workload of the Operating System for receiving and sending.

The chart in the lower half of Fig. 5 shows the timing of the execution of the threads for the CPU usage graph. In this chart, there are twelve rectangles. The width of each rectangle represents the processing period of a thread. As mentioned above, the HS-DRT encryption core module consists of three threads, the receiver thread, the encryption/scrambling thread, and the sender thread. In this chart, the four rectangles that are marked “R#” show the

Table III. practical performance of HS-DRT encryption core module

Processed Data size [GB]	Sending Time [sec]	Processing Time [sec]	Latency [sec]	Performance [MB/sec]	
	PC2	PC1		PC2	PC1
1	9.10	22.0	12.9	113	46.5
2	18.2	31.6	13.4	112	64.8
4	36.5	49.7	13.2	112	82.4
8	73.0	86.2	13.2	112	95.1
12	109	123	13.1	112	100
16	146	159	13.0	112	103
24	219	232	13.2	112	106
32	292	305	13.2	112	107
48	438	451	13.2	112	109

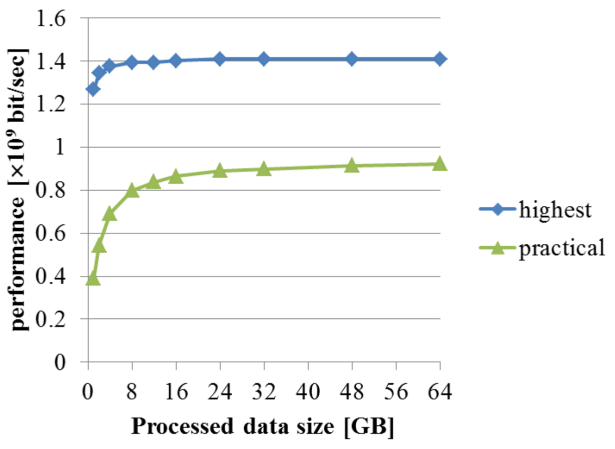

Figure 4. Comparison of throughput for PC1

processing period of the receiver threads, the four rectangles marked "E#" show the processing period of the encryption/scrambling threads, and the four rectangles marked "S#" show the processing period of the sender threads.

For example, the receiver thread received the first 1GB data from the network during "R1" period. Then the receiver thread passed the received 1GB data to the encryption/scrambling thread. The encryption/scrambling thread encrypted the first 1GB data during the "E1" period, and passed this 1GB of data to the sender thread. Finally, the sender thread sent the first 1GB data to the network during the "S1" period. So, the "R1", "E1", and "S1" rectangles form a row.

Looking at the process from a different perspective, the receiver thread received the first 1GB data from network during "R1" period, and passed the received data to the encryption/scrambling thread. Then, the receiver thread received the second 1GB data from network during the "R2" period, and so on.

In the processing period of the encryption/scrambling threads, the "USER" area of the CPU usage graph remains at about 25%. In contrast, the first (leftmost) processing period of the receiver thread and the last (rightmost) processing period of the sender thread consist of "SOFTIRQ" and "SYS" in the CPU usage graph.

From this CPU usage graph and the execution timing chart, it can be seen that the I/O ("SOFTIRQ" and "SYS") of the receiving and sender threads form the main bottleneck of the HS-DRT encryption core module. The proposed encryption core module can achieve the wire speed of gigabit Ethernet with half of 2.4GHz Quad-Core CPU power. So, the proposed method can realize a low cost encryption system with a commercially available cheap Dual-Core CPU.

We also examined the MPEG-2 (8Mb/s) video data transmission performance of the HS-DRT by using a 2.4 GHz Core2 Quad processor. In this case we assumed that the number of GOPs (Group Of Picture) is 15, and 30 frames/s video data transmission (which is equivalent to 512 KBytes per GOP) is imposed. We confirmed that the required encoding time including the encryption and spatial scrambling is 3ms, and the required decoding time including the decryption and descrambling is also 3 ms. This means that the total processing time of 6 ms, which corresponds to the total transmission latency, is sufficiently small and is within the specified standard field time (for example 16 ms).

V. Practical experimental system

In this section, we describe an application for secure video streaming and an application example of the HS-DRT core module used with a cloud computing system.

A. Application of HS-DRT for secure video streaming

HS-DRT is a versatile technology for secure data transmission. One of the network applications of HS-DRT is a secure video streaming service. Figure 6 shows an implementation example of a secure camera monitoring system using HS-DRT. In the streaming sender (on the left-hand side of Fig. 6), the picture frames or GOP of the MPEG video from the camera are encrypted, scrambled, and divided into "m" pieces. Moreover, each of the "m" pieces is further divided into "n" fragments. These (m×n) fragments are sent via (m×n) TCP/UDP streams. At the time of sending, each fragment is assigned the appropriate destination TCP/UDP flow port number by using the shuffled table. In the streaming receiver (on the right of Fig. 6), the "m" pieces are assembled by sorting and merging (m×n) fragments. The receiver merges and descrambles them in the reverse order,

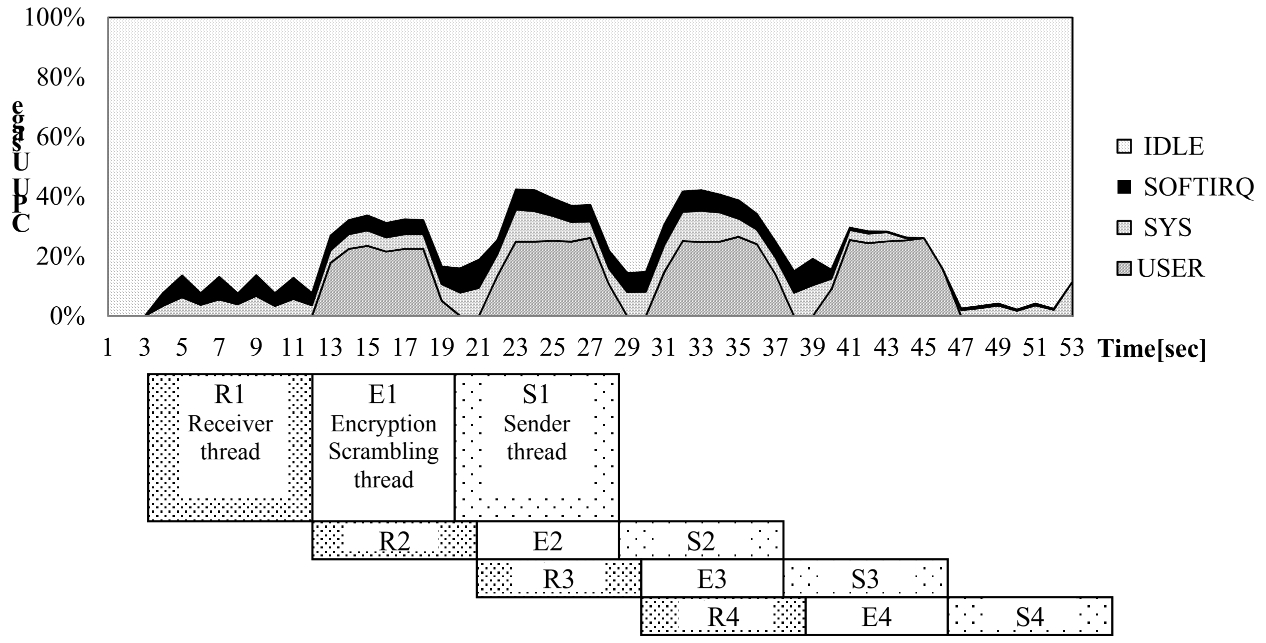

Figure 5. CPU usage on PC1 under practical performance conditions with 4GB data and thread execution timing chart

and decrypts "m" pieces to recover the original captured data. This implementation is considered to be a special case, because the grid nodes and the supervisory server was integrated in the streaming receiver on the right. In this implementation, the sending side and the receiving side have to share secret keys. These shared secret keys consist of the encryption key, the information regarding fragmentation, and the shuffle tables.

B. *Integration with a cloud computing system*

1) *System Implementation*

Figure 7 shows the configuration of one of the practical experimental systems to realize a hybrid HS-DRT processor by making use of a cloud computing system at the same time. As shown in Fig. 7, the system mainly consists of the following four parts: thin clients, a web applications server (Web-Apps Server), an HS-DRT processor, and Storage Clouds. Thin Clients are terminals which can use the web applications in the SaaS (Software as a Service) environment. Thin Clients can make use of the application services which

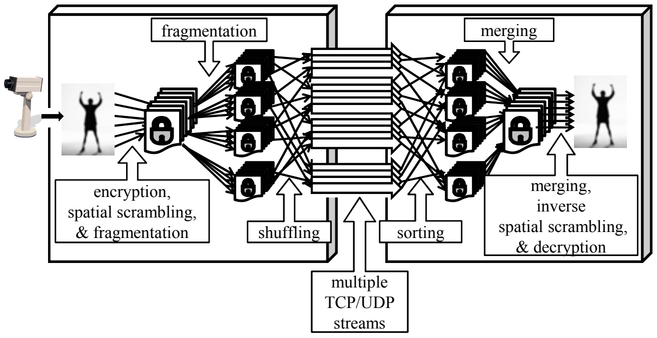
fragmentation
merging
encryption, spatial scrambling, & fragmentation
shuffling
sorting
merging, inverse spatial scrambling, & decryption
multiple TCP/UDP streams

Figure 6. Secure video streaming with HS-DRT

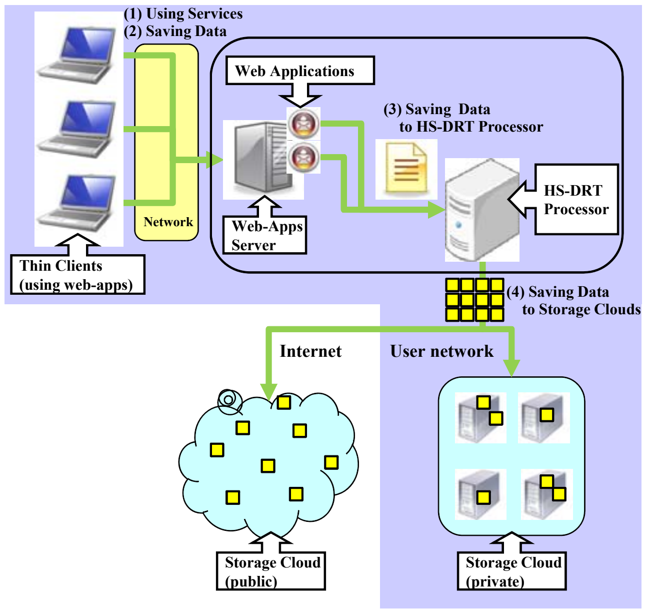

Figure 7. System implementation of HS-DRT processor with cloud computing system

are provided by the web applications server. The HS-DRT processor is considered to be a component of the hybrid cloud computing system, which can also strengthen the cloud computers' security level at the same time. The data center and the supervisory server can be integrated in the HS-DRT processor. The HS-DRT processor can effectively utilize the storage clouds as grid node facilities.

For example, when a user wants to make use of the Web application function for storing specific individual items of data automatically, the user selects the corresponding function, such as "store", or "automatic store", as shown in (2) in Fig. 7.

When the user wants to store the data using the specific HS-DRT processor, the corresponding operation can also be handled by the corresponding user operations, by using well defined GUI operations as shown in (3) in Fig. 7.

In the HS-DRT processor, the HS-DRT-engine, which also has a function related to a web application, executes the encryption, spatial scrambling, and fragmentation of the corresponding files. It sends the corresponding encrypted data fragments to the public or private storage cloud computing system, where they are stored. The choice of whether to use a private cloud or a public cloud follows pre-determined criteria depending on the type of the web application.

2) *HS-DRT Processor structure*

Figure 8 shows the data processing model inside the HS-DRT processor. In Fig. 8, the HS-DRT processor is connected to web applications server on the left, and connected to some cloud computing storage systems on the right. The saved data flows from left to right via the HS-DRT encryption core in the upper half of the HS-DRT processor, and the user data provided in response to a load request flows from right to left via the HS-DRT decryption core in the lower half of the HS-DRT processor.

The data from the web application is first received at the engine, which is equipped with a distributed cache. This cache enhances the response time of the HS-DRT processors to the web applications server in both saving and loading operations. The data which is to be stored is sent to the HS-DRT encryption core and thereafter, encrypted, spatially scrambled, and divided into fragments. Following this, the HS-DRT distributes the data by transmitting the fragments.

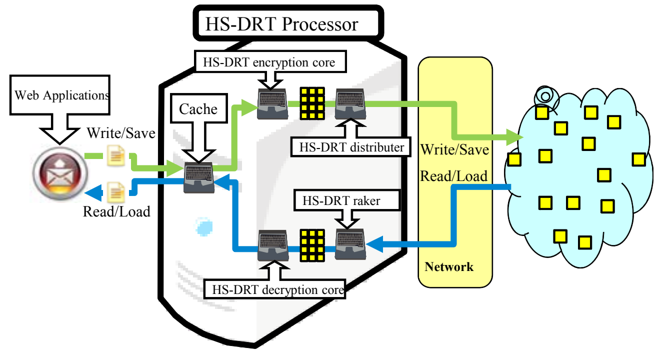

Figure 8. HS-DRT Processor structure

When there is a request from the web applications to load data, the cache engine first processes the data. The HS-DRT decryption core engine indicates that the HS-DRT-raker should collect back the distributed data from the relevant storage clouds. The engine then sends the collected data, which can be used for file data recovery if required.

3) Characteristics of HS-DRT processor integrated with the cloud computing system

It is very important to note that the processing efficiency of the HS-DRT processor can easily be improved by increasing the amount of the web cache memory.

We need to consider the scalability of the engine, since it may become a bottleneck in a very large system, owing to the number of clients and the amount of storage. In such cases, the HS-DRT processor may use a key-value database. As the HS-DRT processor can easily work with other HS-DRT processors, the system can be extended. The secrecy of the system can easily be assured because there is no plain raw data stream appearing anywhere in the entire data processing procedure.

On the other hand, we should take the following disadvantageous points into account.

- In order to fully utilize the HS-DRT processor, the web applications need to be well adjusted to be used by the HS-DRT engine itself.
- If the number of replicated copies of file data increases, the corresponding processor performance for executing the web applications will be degraded accordingly.

As for the HS-DRT processor, it can include the function of the web application server in addition to the encryption, spatial scrambling, and fragmentation of the corresponding file data. Scalability can also be easily attained by making use of conventional technologies for load balancing and multiplexing of the processor.

VI. CONCLUSIONS

We have presented an experimental evaluation of the encryption and the spatial scrambling performance of the proposed data recovery system, and found that the average response time in terms of the file data size is sufficiently practical to realize the corresponding network services.

Discussion has also been provided on an effective shuffling algorithm using Mersenne Twister and "Fisher-Yates shuffle" to determine the dispersed location sites.

Finally, this paper has described a prototype system configuration for several practical network applications, in particular, implementing a hybrid structure by making use of cloud computing facilities and environments which have already been commercialized.

Further studies should address the optimum network utilization technology. We are planning to verify the essential characteristics necessary to fully utilize the network resources to commercialize an ideal disaster recovery system.

ACKNOWLEDGMENTS

This work has been partially supported by the Grants-in-Aid for Scientific Research (Issue number: 21560414), the Research Institute for Science and Technology of Tokyo Denki University (Q09S-10) and a grant of Strategic Research Foundation Grant-aided Project for Private Universities from Ministry of Education, Culture, Support, Science, and Technology, Japan (MEXT), 2007-2011.

REFERENCES

[1] Y. Ueno, N. Miyaho, S. Suzuki, and K. Ichihara, "Performance Evaluation of a Disaster Recovery System and Practical Network System Applications," ICSNC 2010, pp. 195-200, Aug., 2010.

[2] N. Miyaho, Y. Ueno, S. Suzuki, K. Mori, and K. Ichihara, "Study on a Disaster Recovery Network Mechanism by Using Widely Distributed Client Nodes," ICSNC 2009, pp. 217-223, Sep., 2009.

[3] S. Suzuki, "Additive cryptosystem and World Wide master key," IEICE technical report ISEC 101(403), pp. 39-46, Nov., 2001.

[4] N. Miyaho, S. Suzuki, Y. Ueno, A. Takubo, Y. Wada, and R. Shibata, "Disaster recovery equipments, programs, and system," Patent. publication 2007/3/6 (Japan), PCT Patent :No.4296304, Apr. 2009.

[5] Y.Ueno, N.Miyaho, and S.Suzuki, "Disaster Recovery Mechanism using Widely Distributed Networking and Secure Metadata Handling Technology," Proceedings of the 4th edition of the UPGRADE-CN workshop, Session II: Networking, pp. 45-48, Jun., 2009.

[6] K. Kokubun, Y. Kawai, Y. Ueno, S. Suzuki, and N. Miyaho, "Performance evaluation of Disaster Recovery System using Grid Computing technology," IEICE Technical Report 107(403), pp. 1-6, Dec., 2007.

[7] NTT-East "Wide area disaster recovery services", <http://www.ntt-east.co.jp/business/solution/security/dr/> 23.05.2010

[8] Y. Kitamura, Y. Lee, R. Sakiyama, and K. Okamura, "Experience with Restoration of Asia Pacific Network Failures from Taiwan Earthquake," IEICE Transactions on Communications E90-B(11), pp. 3095-3103, Nov. , 2007.

[9] S. Kurokawa, Y. Iwaki, and N. Miyaho, "Study on the distributed data sharing mechanism with a mutual authentication and meta-database technology," APCC 2007, pp. 215-218, Oct., 2007.

[10] J. Yamato, M. Kan, and Y. Kikuchi, "Storage Based Data Protection for Disaster Recovery," The Journal of the IEICE 89(9), pp. 801-805, Sep. , 2006.

[11] Shanyu Zhao, Virginia Lo, and Chris GauthierDickey, "Result Verification and Trust-Based Scheduling in Peer-to-Peer Grids," p2p, pp. 31-38, Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05), 2005.

[12] K. Sagara, K. Nishiki, and M. Koizumi, "A Distributed Authentication Platform Architecture for Peer-to-Peer Applications," IEICE Transactions on Communications E88-B(3), pp. 865-872, 2005.

[13] S. Tezuka, R. Uda, A. Inoue, and Y. Matsushita, "A Secure Virtual File Server with P2P Connection to a Large-Scale Network," IASTED International Conference NCS2006, pp. 310-315, 2006.

[14] R. Uda, A. Inoue, M. Ito, S. Ichimura, K. Tago, and T. Hoshi, "Development of file distributed back up system," Tokyo University of Technology, Technical Report, No.3, pp. 31-38, Mar 2008.

[15] Y. Deswarte, L. Blain, J.-C. Fabre, "Intrusion tolerance in distributed computing systems," Research in Security and Privacy, 1991. Proceedings, 1991 IEEE Computer Society Symposium, pp.110-121, 20-22 May 1991

[16] R. A. Fisher and F. Yates, Statistical tables for biological, agricultural and medical research (3rd ed.). London: Oliver & Boyd. pp. 26–27, 1948.

[17] D. E. Knuth, The Art of Computer Programming, Volume 2: Seminumerical algorithms., 3rd edition, Addison Wesley. pp. 142-146, 1998.