Journal of Information Security, **2013, 4, 42-53**
http://dx.doi.org/10.4236/jis.2013.41006 Published Online January 2013 (http://www.scirp.org/journal/jis)

Technology of Secure File Archiving in the Uniformly Random Distributed Archive Scheme

Ahmed Tallat, Hiroshi Yasuda, Kilho Shin
Applied Information Engineering, Tokyo Denki University, Tokyo, Japan
Email: taltdulat@gmail.com

Received September 29, 2012; revised October 30, 2012; accepted November 10, 2012

ABSTRACT

This paper investigates the security features of the distributed archive scheme named Uniformly Random Distributed Archive (URDA). It is a simple, fast and practically secure algorithm that meets the needs of confidentiality and availability requirements of data. URDA cuts a file archived into fragments, and distributes each fragment into randomly selected $n - k + 1$ storages out of n storages. As the result, users only need to access at least k storages to recover original file, whereas stolen data from $k - 1$ storages cannot cover original file. Thus, archived files are nothing but sequences of a large number of fixed length fragments. URDA is proved of disappearing both characters and biased bits of original data in archived files, indicating the probabilities of both a fragment and a bit appearing at particular position are uniformly constant respectively. Yet, through running experiments, we found out the risk of likelihood that URDA might be vulnerable to bit pattern attack due to the different ratios of characters appearing in real world files. However, we solved the problem by modifying URDA with variable fragment lengths, which results in that all the bits in revealed sequences are distributed uniformly and independently at random.

Keywords: Cloud Computing; Archive; Secret Sharing; Character Set

1. Introduction

Cloud computing is attracting attentions of people of a wide variety of fields including users, service providers and facility manufacturers. From the users' point of view, the most attractive advantages of cloud computing must be the ubiquity and affordability of services that cloud computing can realize. Since cloud computing is constructed on top of the Internet, users can take advantage of services anytime and anywhere. The recent rapid spread of mobile computing certainly enhances the demands for the ubiquity of service provision. On the other hand, the affordability of cloud computing is based on exploitation of public infrastructures. The Internet itself is apparently public good, and cloud computing is enlargeing its scope to computational powers and storages that are available in the Internet. For example, the data backup services provided by data center companies are very expensive, and small and middle-sized enterprises cannot afford them. In contrast, the same services are now being provided at significantly lower prices based on cloud computing.

Thus, the advantages of cloud computing are due to the fundamental idea of exploiting the Internet and public goods existing in the Internet. This, however, can produce disadvantages as well. In particular, in terms of security, the Internet has serious problems. ISO 17799 [1] determines that confidentiality, integrity and availability (CIA) are the three important factors in considering security issues, and the Internet has vulnerability in all of them: Communication over the Internet is in principle subject to eavesdropping and tampering; Nobody controls the overall the services of the Internet, and cutoff and delay of communication can happen anytime. Furthermore, we cannot deny the possibility that the providers of services are malicious. Although cryptographic technology will provide us with strong countermeasures with respect to confidentiality and integrity, we need different technologies to solve the issue of availability.

Let's consider a data backup service based on cloud computing. A user uploads his or her data to the Internet for the backup purpose, and the data are distributed across many inexpensive storages unknown to the user. Different storage providers operate different storages under different policies. Even their connectivity may vary. Thus, the user can have a problem when he or she needs to restore the data. In the worst case scenario, it is still possible that some of storage providers have ceased their services without notification, and therefore, a part of the backup data has been lost forever.

To solve this problem of availability, storing data with redundancy is probably the only solution, and we have

already had technologies to realize availability by redundant storing.

RAID [2] (Redundant Array of Inexpensive Disks) is daily use technology, and combines more than one physical hard disks into a single logical unit by distributing the whole data mostly with redundancy across multiple disks, which is necessary in case of data lost; A failed disk is replaced by a new one, and the lost data will be built from the remaining data and the redundancy such as parity data. This technology enables computer users to achieve high level storage reliability from PC-class components.

Although RAID is appropriate to apply to "home" servers, it has evidently issues in its scalability. In particular, RAID can accept failures of only one (RAID 3 and 5) or two disks, and will not be able to apply to the Internet, where users may lose access to more than two storages.

The technology known as "secret sharing" also provide a method to distribute secret information with redundancy: For $n > k > 0$, n members have their shares derived from a secret, and the secret will be completely reconstructed when k members exhibit their own shares; By contrast, collusion of $(k - 1)$ members will reveal none or only a few bits of the secret. This property of secret sharing is referred to as k-out-of-n threshold secrecy. The initial (k, n) threshold secret sharing scheme (SSS) known in the literature were independently invented by Adi Shamir [3] and George Blackley [4] in 1979, and since then the scheme has been widely studied and applied on a variety of fields. (for example, Rabin [5], Bai [6,7], Blundo [8], He [9], Wang [10]).

Although these schemes support the severe requirement of k-out-of-n threshold secrecy, their computational complexity is extremely high, and it cannot show practical performance when applied to archive of bulky data: data should be fragmented into a number of fragments so that they are short enough to be dealt with by the secret sharing algorithms, and the algorithms need heavy computation such as modular exponentiation and matrix manipulation in calculating shares from each fragment.

The recent technology of P2P also provides the function of storing data with redundancy. P2P aims at sharing decentralized resources (CPU, storage, bandwidth and contents) of each participant, while eliminating conventional centralized unit to provide browsing and downloading, and thus the participants are both consumer and provider of the resources in distributed networks. The principle is simply uploading and downloading simultaneously and continuously, and the acceleration of data transferring can be improved significantly by downloading desired pieces simultaneously with random order from multiple providers that treat distributed contents as sequence of pieces through fragmenting them into sequences. We already have several P2P based global storage products (for example, OceanStore [11] and PAST [12]).

The most important problem of the P2P technology when applying to our purpose should be its extremely low efficiency in storage spaces and bandwidth. The initial aim of P2P is to realize sharing data among an unspecified number of people, and hence, data are exchanged and stored with very high redundancy exceeding the necessity for the backup purpose.

In our previous work [13], we investigated a simple scheme, called the uniformly random distributed archive scheme (URDA). The algorithm of URDA is extremely simple and easy: The original data is fragmented into a number of tiny fragments (several bits long), and each fragment is distributed across $n - k + 1$ storages out of n storages: The most important feature of URDA consists in that the selection of the $n - k + 1$ storages is dominated by a completely random process: Storages are selected uniformly and independently at random per fragment. By this, URDA meets the condition of k-out-of-n threshold robustness: that is, even if the owner of the data loses access to $k - 1$ storages of the entire n storages, the user can restore the original data from the data fragments retrieved from the remaining k storages.

In terms of confidentiality, it has been revealed in previous work that URDA has a couple of remarkable properties.

1) Except for bits existing in the very narrow neighborhood of the start and the end of the original file, the probability that a guess on the position where a particular fragment of the original file appears in a distributed backup file is very small, and almost constant.

2) When assuming that the original data consists of ASCII characters selected at random from the entire possible values, the probability that a particular bit pattern appears at a particular position in a distri- buted backup file is independent of the position. To be precise, we can theoretically derive the following formula.

$$\Pr\left[b_t = \cdots = b_{t-l} = 0\right] \approx \frac{1}{8}\left(\frac{l+1}{2^l} + \frac{7-l}{2^{l+1}}\right) = \frac{l+9}{2^{l+4}} \qquad (1)$$

The symbol b_t represents the t-th bit from the head of the backup data, and the formula holds unless the bit falls into narrow neighborhoods of the head and the tail of the distributed backup file.

The first property implies that, if attackers do not take advantage of biases of bit patterns in the original data, they cannot guess the contents of the original data from the distributed backup data. On the other hand, in the second property, we take ASCII data as an instance, and show that the evident bias of bit patterns, that is, 0 ap-

pears every eight bits, disappears in distributed backup files. Thus, these properties indicate the possibility that URDA can provide confidentiality of a certain degree, without relying on cryptographic techniques. This would be an important advantage of URDA in terms of time-efficiency of the scheme.

In this paper, we further investigate the second property. In the real world, any text files have particular biases in the distributions of bit patterns, and the theoretical conclusion of URDA might not hold true for data in the real world. We struggle with this problem through experiments.

In the experiments, we downloaded 200 ASCII files from the Internet, applied URDA to these files to generate 1000 backup files, and looked into the statistical features of the backup files. First, we found that the Equation (1) still holds true for the backup files we investigated in our experiments. This was verified by performing the one-sample *t*-test: It has turned out that the "null" hypothesis that the Equation (1) does hold cannot be rejected even with a large significance level. Secondly, we synthetically generated 200 random ASCII files and 1000 backup files in the same way as for the downloaded ASCII files. Then, we compared between the synthetic and the real backup files in terms of the variance of the distributions of the particular bit pattern. The result this time showed that the null hypothesis can be rejected with the significance level 0.05 by the F-test. This means that the distributions of bit patterns between the real and synthetic backup files are different, and therefore, we cannot deny the possibility that a clever attacker can invent an effective pattern analysis to guess the original contents from the contents of the backup files.

Based on the result of this experiment, we modified the algorithm of URDA so that the length of fragments are to be determined at random per fragment, and performed the same experiment using this modified URDA. The result was surprising. The two groups of distributions, one for the real backup files generated by the modified URDA and the other for the synthetic backup files, are concluded to be the same. To be precise, even with a large significance level, we could not reject the null hypotheses that claim that the distributions are the same as Gaussian distributions. This consequence is significant: The backup files of real ASCII files generated by the modified URDA are indistinguishable from the synthetic backup files. By definition, the synthetic backup files show the uniform randomness in terms of bit pattern distributions, and hence, we can conclude backup files generated by the modified URDA are secure against bit pattern analysis.

Following Section 2 describes some related studies followed by URDA scheme proposed in previous paper for the consistency of the paper, and it also includes analyses of security feature of URDA. Section 3 clarifies problem that the paper is tackling, and provides solution by running experiments and applying statistical methods. We conclude the paper with future work in section 4.

2. Distributed Archive Schemes

In this section, we provide a brief review over the data storage techniques of redundancy known in the literature, and then describe URDA proposed in our previous work.

2.1. Distributed Schemes in Storage Pools

Digital high density recording technologies have made a variety of data recorded in storage medium. As a result, the form of utilizing storage devices has been greatly changed, and a variety of home appliances, computers/servers, and even micro device in mobiles have become to have their own storages and to utilize them at a higher level.

However, with the advent of the Internet the need of sharing information made rapid increase of data in terms of variability and amount, which in turn led to creation of high technical processing power of computers and also posed strong demand to the scale of flexibility of storages for these ever-increasing data. Since there are limits of computational power and capacity within conventional devices having attached to individual storages inside, they are no longer able to meet the need the way of data being accessed, stored and shared. Furthermore, tracking and backing up files distributed into a variety of storages constitute significant hardship. Some storage techniques [14] try to address these problems by allowing all the files to be stored in a single, secure storage that can be accessed by other clients and servers regardless of the operating systems from anywhere within the same domain.

However, independency of storages and servers is costly in terms of updating, managing and running. Thus, the conception of "storage pool" [15] aims at providing the scale of flexibility to solve the problems by allowing servers to utilize storage capacity from the pool so that virtual hard disk drives can be dedicated to servers based on their needs without buying extra storage to each of them individually. The storage pool is also known as a dedicated storage network, because it is separated and independent from servers, allowing several servers to connect one drive and vise verse. Same as the independent storage, drive enclosure in the storage pool can hold any number of drives with a variety of types and expand on demand. Mostly the variety of drives has a central control unit to manage all the Input/output and they are equipped with some technical schemes for providing security and recoverability in case of disaster or system failure.

However, to guarantee the availability, integrity and confidentiality of the data stored in storage medium is not easy task. Because every system is vulnerable to certain damages caused by natural, physical and technical attacks, and thus these data can be lost, delayed or even stolen. Redundancy probably is the only feasible solution to address the issues. The fastest method to realize redundancy is replication [16], which distributes redundantly copied fragments of data across infrastructure, but it is space inefficient, and p2p is a typical example of the replication-based systems. Another resilient method is erasure coding that splits data into n fragments, which are then further redundantly encoded into k additional fragments using parity, matrix, polynomial etc. Thus, k specifies the level of resiliency or Maximum Distance Separable (MDS), where k = MDS.

Despite their values, widespread utilities, and reliabilities due to the capabilities of meeting the various needs for disaster recoveries, they all have drawbacks in terms of heavy cost and complex computational procedures. Because they use matrices, cryptography and heavy mathematical calculation etc. for producing redundancy, and applying them to large amount of data is very expensive, due to extremely time-consuming computational requirements, and even they are relatively inflexible, due to the complexity and difficulty of meeting various levels of desired security requirements as needed. What is more, dealing with cloud computing environment, for example for the purpose of backup of datacenter, is not easy with these schemes, due to lack of utilization of cost-effect readiness to use available storages provided. Thus, the currently heavy set-up cost and computational complexity cause significant hardship to utilize currently available storage services in order to realize secure backup services.

2.2. URDA-Uniformly Random Distributed Archive Scheme

In order to address aforementioned issues of current techniques when applying them to cloud computing in the context of robustness and security, and to provide affordable backup techniques, we have introduced URDA in our previous work.

The outline of the algorithm of URDA is as follows: URDA first fragment the source file to archive into multiple tiny fragments, and then distributes each fragment to $n - k + 1$ destination storages selected out of the entire n storages. The $n - k + 1$ storages are selected uniformly and independently at random per fragment. Thus, the source file is distributed across n storages, and each fragment is archived at exactly $n - k + 1$ different storages. The simple image of URDA is shown in **Figure 1**.

To be specific, URDA consists of a smartcard and a host (e.g. client/server software), and they communicate with each other through a standardized interface like IC card interface [17], Near Field Communication [18] and Infrared Link Access Protocol [19]. The role of the smartcard is to generate random distribution keys taking advantage of the key generator RNG_K receiving seeds from the seed generator RNG_S, both RNG_K and RNG_S are installed inside the smartcard, whereas the host fragments the source file, duplicates the fragments, and distributing the copies generated across multiple storages, following the indication by the random distribution keys. In the following, we give a brief description of the archival and retrieval phases of URDA. The flow chart of the algorithm is shown in **Figure 2**.

(1) *Archival phase*: The role of the smart card in the archival phase is to generate a series of distribution keys and to transmit them to the host. The algorithm of the smart card in the archival phase is as follows;

1) Receive the START_ARCHIVAL signal accompanying the identifier of the source file through the card-host interface. The identifier is denoted by crr_fid.

2) Generate a random seed taking advantage of RNG_S. The seed is denoted by crr_seed.

3) Store the pair of crr_fid and crr_seed in the internal

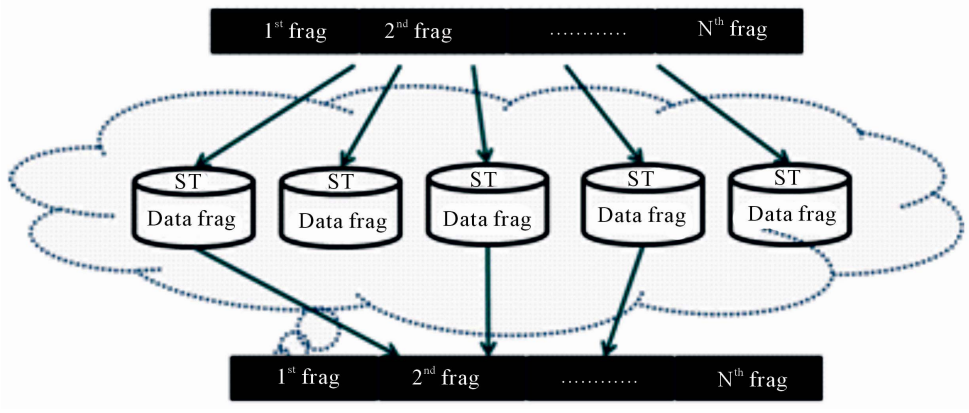

Figure 1.The simple image of URDA (n = 5, k = 3).

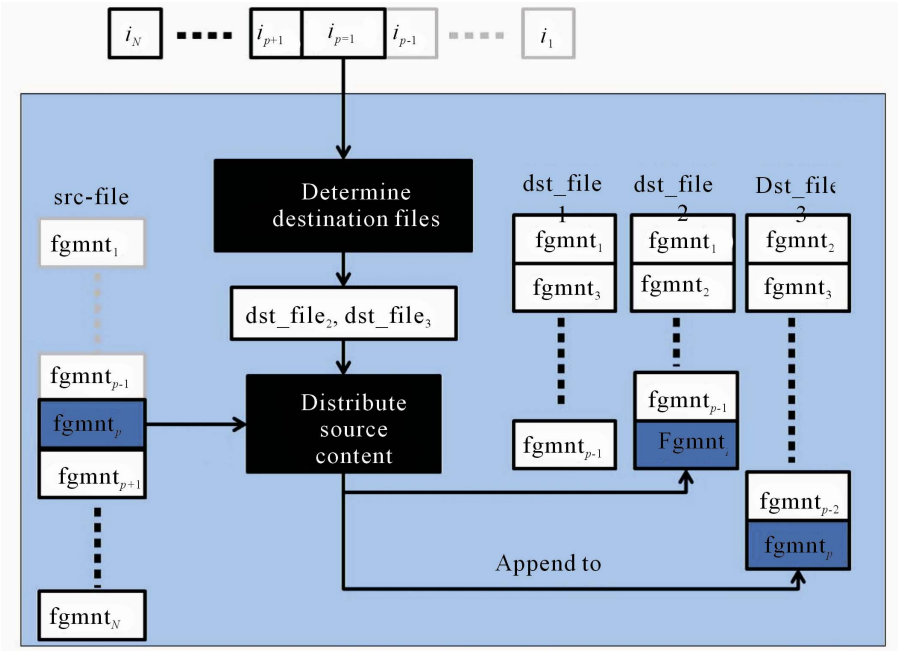

Figure 2. The flow chart of URDA.

database. The pair will be used in the retrieval phase.

4) Input crr_seed into RNG_K.

5) Let RNG_K generate a random number, which is an index to determine the (n; k)-distribution key to be used by the host. The smart card outputs the random number to the host program.

6) Repeat Step 5 until it receives the END_OF_FILE signal from the host program.

On the other hand, the role of the host program in the archival phase is to distribute the contents of the source file over n destination files.

1) Fragment the source file into the b-bit long fragments of $fgmnt_1,\cdots,fgmnt_{last}$, and create a pointer p to indicate the current fragment $fgmnt_p$. The initial value of p is 1.

2) Send the START_ARCHIVAL signal to the smart card.

3) Receive an index i_p from the smart card.

4) Identify the distribution with index i_p. The distribution key determines the set of $n - k + 1$ destination files to which the current fragment $fgmnt_p$ is to be appended.

5) Take away the current fragment $fgmnt_p$ from the source file, and increment the pointer p by 1.

6) Append the fragment $fgmnt_p$ to the destination files determined in Step 4.

7) Repeat the steps from 3 to 6 until all the fragments are taken away from the source file.

8) Send the END_OF_File signal to the smart card.

9) The host program sends the resultant n destination files to n storages via networks.

(2) *Retrieval phase*: The algorithm of the smartcard in the retrieval phase is as follows:

1) Receive the START_RETRIEVAL signal accompanying the identifier crr_fid of the target file from the host program.

2) Look up crr_fid in the internal database, and retrieve the seed crr_seed paired with crr_fid.

3) Input crr seed into RNGK.

4) Output the random numbers i_p that RNGK generates to the host program.

5) Repeat Step 4 until the END_OF_FILE signal is received.

On the other hand, algorithm of the host is described as follows.

1) Fragment the destination files into b-bit length fragments, create a pointer for each destination file. Initially, each pointer is set so that it indicates the first fragment.

2) Send the START_RETRIEVAL signal to the smart card.

3) Receive an index i_p from the smart card. The initial value of p shall be 1.

4) Determine the set of $n - k + 1$ destination files in which the current fragment $fgmnt_p$ are included.

5) For each of the destination files determined in Step 4, take away the fragment indicated by the pointer associated with the destination file, and change the pointer so that it points the next fragment.

6) Append $fgmnt_p$ to the source file, and increment p by 1.

7) Repeat the steps of 3 to 6 until all of the destination files are empty.

The size of the resultant destination files is exactly $(n-k+1)$ times as great as the original size of the

source file and the expected size of each destination file is $(n-k+1)/n$ times smaller than the size of the source file, and thus it supports (k,n) robustness and (k,n) security, which means that downloading from k storages is enough to recover original file and that even $(k-1)$ storages are attacked, the stolen data cannot cover total amount of the file.

2.3. Security Features of URDA

In this clause, we summarize the security features of URDA that are presented in our previous work.

The probability that a specific fragment is not included in any of specific r storages is

$$\frac{n-k+1}{n} * \frac{n-k}{n-1} * \cdots * \frac{n-k-r}{n-r-1}$$

Thus, if r storages are compromised, the probability that all of the fragments are revealed is

$$\left(1-\prod_{i=1}^{r}\frac{n-k-i}{n-i-1}\right)^{N} \quad (2)$$

N is the total number of the fragments. When $r \geq k$, the Equation (2) is 1, whereas, when $r < k$, it approaches 0, as N increases. This implies the k-out-of-n threshold secrecy: Even if $k-1$ storages are compromised, the entire original contents will not be revealed.

Definition 1. Let N and m denote the number of fragments in a source file and a revealed sequence, and a and j denote the positions of a fragment in both the source file and the revealed sequence $(N \geq a \geq 1, m \geq j \geq 1)$ respectively. Then we define that $\mathbf{Match}(a, j; N, m)$ denotes the probability that a fragment at the position of a in the source file appears at the position of j in the revealed sequence.

When a is fixed and j moves, following holds for the maximum value of $\mathbf{Match}(a, j; N, m)$.

$$\underset{j\in\{1,\cdots,m\}}{\mathbf{argmax}}\,\mathbf{Match}(a, j; N, m) = \left\{\left\lceil \frac{am}{N+1}\right\rceil, \left\lfloor \frac{am}{N+1}\right\rfloor + 1\right\}$$

Thus, the maximum of $\mathbf{Match}(a, j; N, m)$ when a is fixed and j moves, is given by the following formula.

$$\mathbf{Match}\left(a, \frac{am}{N+1}; N, m\right) \approx \frac{m}{N\sqrt{\pi}} \cdot \left(\frac{N-1}{(a-1)(N-a)}\right)^{1/4} \quad (3)$$

The minimum value of the equation can be reached at $a = \frac{N-1}{2}$, and **Figure 3** is its simulation result for the value of $N = 10^3, 10^4, 10^5, 10^6$ and $\frac{m}{N} = \frac{3}{4}$.

Thus, we can conclude that the security against identifying the position of a fragment is uniformly small except the fragments located in heads and tails. Furthermore, obtaining further safety can be possible by adding dummy data at the heads and tails.

Next, we run a thought experiment. In the experiment, we see that the certain significant pattern of the bit distribution of ASCII text files disappears in archived backup files. For the ASCII text files to use in the experiment, we assume that ASCII characters in the files are selected uniformly and independently at random per character from the entire possible values. Therefore, the bit distribution of the files is as follows: The most significant bit (MSB) of every byte is always 0, whereas the remaining bits are distributed uniformly and independently at random. Then, we define

$$\bar{q}(j, \rho) = \sum_{a \bmod 8 = \rho} \mathbf{Match}(a, j; N, m)$$

to represent the probability that $a \bmod 8 = \rho$ holds when fgmnt_j is a copy of fgmnt_a. Then, the probability that the ρ-th bit $(\rho = 1, \cdots, 7)$ of fgmnt_j is 0 turns out to be

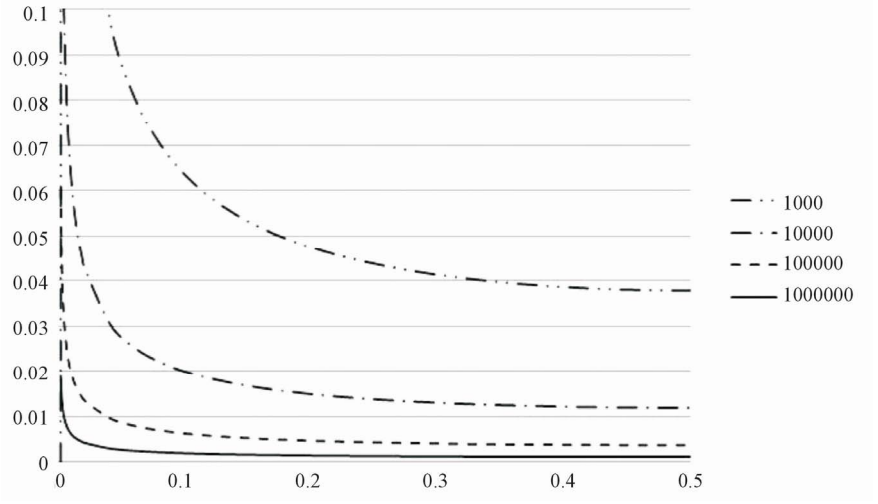

Figure 3. Simulation result

$$\Pr\left[b_{7j+\rho}=0\right]=\overline{q}\left(j,\rho\right)+\left(1-\overline{q}\left(j,\rho\right)\right)\cdot\frac{1}{2} \qquad (4)$$

A remarkable fact is that $\overline{q}\left(j,\rho\right)\approx\frac{1}{8}$ holds except for narrow neighborhoods of $j=1$ and $j=m$. Hence, when *t* is not close to 1 or *m*, we have the following approximation.

$$P\left[b_t=\cdots=b_{t-l}=0\right]\approx\frac{1}{8}\left(\frac{l+1}{2^l}+\frac{7-l}{2^{l+1}}\right)=\frac{l+9}{2^{l+4}}$$

Through the same computation as above, we have the following.

Corollary 1. When *t* is not close to 1 or *m*, $\Pr\left[b_t=\beta_t,\cdots,b_{t+l}=\beta_{t+l}\right]$ is approximately constant.

Corollary 2. When *t* is not close to 1 or *m*, $\Pr\left[b_t=\beta_t,\cdots,b_{t+l}=\beta_{t+l}\right]$ is approximately constant.

Corollary 1 and 2 imply that most of the bits in the revealed contents look almost the same in the attacker's eye.

3. Problem and Solution

In following sections, through identifying the problem, we provide particular solutions to address it. To proceed, we investigate experimentally on ASCII characters to see if bit patterns of source files can be disappeared in archived files, and further demonstrate that practical security of plain text archiving can be obtain by varying fragment length in bit distribution.

3.1. Identified Problems

Our previous work, assuming that source files include only ASCII characters selected uniformly and independently at random, we mathematically proved that an archived file consisting of seven-bit-long fragments shows uniformly constant distributions with respect to the probabilities that particular bit patterns appear in it. To be precise, we let $C_1\cdots C_L$ be the byte contents of a file, where c_i is an ASCII character selected uniformly at random from the entire space of ASCII characters. Then, we apply URDA to this file by setting the fragment length to be seven bits, and hence, the file content is fragmented into seven-bit-long fragments. The resulting archived files are collections of these fragments. Focusing on one of the archived files, we proved that the probability $\Pr\left[b_{t-\ell}=\beta_{-l},b_{t-\ell+1}=\beta_{-l+1},\cdots,b_t=\beta_0\right]$ is approximately constant regardless of the choice of t, unless t is too small or too large. b_t denotes the bit at the position *t* in the archived file, and $\beta_{-\ell}\beta_{-\ell+1},\cdots,\beta_0$ is an arbitrary bit pattern with $\beta_{-i}\in\{0,1\}$. This means that we can view the archived file as secure against bit pattern analysis, since the source file does not include any bit pattern except that the bit 0 appears every 8 bits, and this pattern proves to disappear in the archived file.

On the other hand, text files found in the real world, even when they consist of ASCII characters, cannot help including biases in the distribution of characters. Thus, the hypothesis of the aforementioned investigation does not necessarily hold, and hence, it is not certain that the conclusion of the investigation holds true for the real files.

The problem that we solve in this paper is first to investigate whether the property that we proved in our previous work also holds true for real files that include biases in the distributions of characters. Then, we will answer the question whether URDA is secure against bit pattern analysis when applied to real files.

3.2. How to Solve the Problem

To solve this problem, we run experiments with a large number of files that we obtain from the internet.

To be specific, we download ASCII files from diverse sites to apply URDA to these files, and then investigate the distributions of bit patterns that appear in the resulting archived files. We first examine whether the mean of the probabilities that particular bit patterns (in the experiments, we focus on four bit patterns that are sequences of 0) is identical to what was theoretically derived in Equation (1).

When we naturally assume that the probabilities observed follow Gaussian distributions, it is not sufficient to investigate only the means. We have to investigate the variances as well. For this purpose, in addition to the files obtained from the Internet, we generate synthetic ASCII files based on the uniformly random process: For each position in a file, an ASCII character is selected uniformly and independently at random. Then we compare the distributions of the probabilities of the same bit patterns between the archived files generated from the downloaded files and those derived from the synthetic files. The comparison will be made in terms of the means and the variances, since we assume that the distributions follow Gaussian distribution. Since the synthetic files are generated based on the same probabilistic model as what Equation (1) was based on, and since we concluded that such files are secure against bit pattern analysis, this comparison shows an answer to the question whether the same conclusion with respect to confidentiality holds true for real ASCII files.

3.3. URDA with the Setting of the Fragment Length = 7 bits

The procedures to run the experiment are given as follows.

1) Find 200 ASCII files in the Internet, and download them.

2) Apply URDA to the downloaded files with $n = 5$ and $k = 3$, and hence, obtain 1000 archived files, each of which is at least 100 KB length. Here, we set the fragment length to be seven bits as specified in Section 2.2.

3) For t = 1 to 400,000, count the number n_t of occurrence of $b_t = \cdots = b_{t-\ell} = 0$ for $\ell = 0,1,2,3$ over the 1,000 archived files, and calculate $\Pr[b_t = \cdots = b_{t-\ell} = 0]$ by $\frac{n_t}{1000}$. Thus, we have 400,000 scores of $\Pr[b_t = \cdots = b_{t-\ell} = 0]$ for each $\ell = 0,1,2,3$ respectively.

We plot these scores in the graphs of **Figure 4**.

Yet we are still unable to estimate whether the mathematical formula can also apply to the real world biased ASCII files with fixed fragment length, without evaluating dispersion of the scores from means.

3.4. Statistical Tests

According to the mathematical evaluation given in URDA in previous work, the scores vary drastically when t is small (the position is close to the head of the files), but they quickly converge to Equation (1).

In fact, when looking at **Figure 4**, the scores observed in this experiment seem to follow this rule: The values of $\frac{l+9}{2^{l+4}}$ are 0.5625, 0.3125, 0.1719 and 0.0938 respectively for l = 0,1,2,3. In the following, we will investigate this, that is, whether the scores observed in fact follow the statistically calculated mathematical formula proved in previous work.

3.4.1. Examining the Means

By the one-sample t-test, we investigate whether the population means underlying the observed scores are identical to the test means calculated by Equation (1).

The null hypothesis here is of course that the population means are identical to the test means. **Table 1** shows the results of one-sample t-test.

The t statistics are calculated from the equation $t = \frac{\bar{x} - \mu}{\frac{s}{\sqrt{n}}}$, where $\bar{x}, \mu, n$ and s stand for the sample mean, test mean, sample size and standard deviation of the sample respectively. From **Table 1**, we see that we cannot reject the null hypothesis even with very large significance level, say 0.2: Typically, we use the signifycance level 0.05 or 0.01. Hence, we can reasonably guess that the null hypothesis is right, that is, the population means are identical to the theoretical value calculated by Equation (1).

3.4.2. Examining the Variances

In the previous clause, we see that the population means are identical to the theoretical means calculated from Equation (1). This result alone, however, is insufficient, since this result tells nothing about how the scores vary around the means. Thus, we have to investigate whether the population variances are sufficiently small. The problem here is that, unlike the calculation of means, we do not have a formula to give the theoretical expectation for the variances. As an alternative method, we generate synthetic files that are generated based on the same probability model as the mathematical model that underlies Equation (1), and compare the scores obtained from the experiment and the scores derived from these synthetic files.

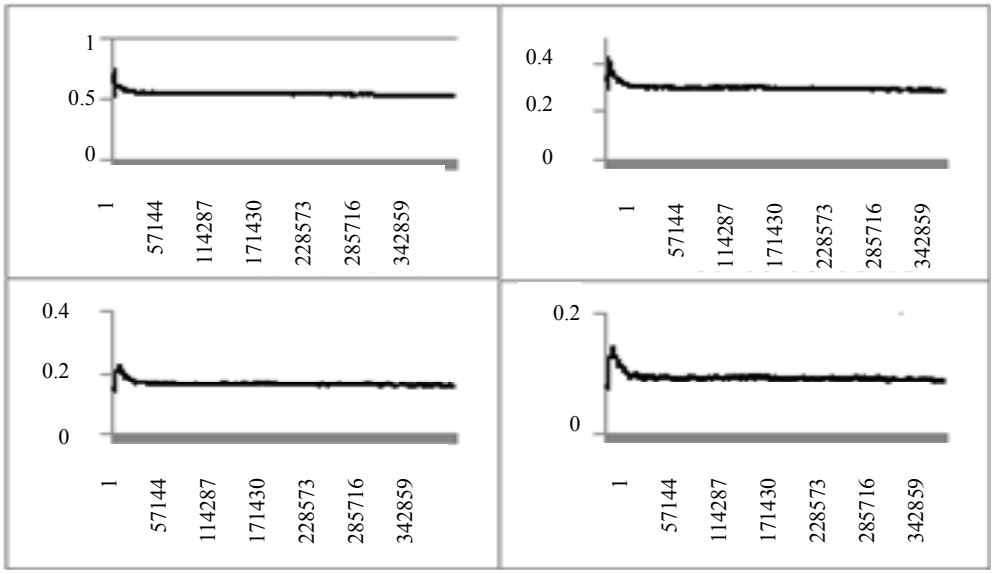

Figure 4. Fixed-fragment-length Pr[$b_t = \cdots = b_{t-l} = 0$] (l = 0,1,2,3, t = 1, ⋯ , 400,000).

Table 1. Comparing of observed scores and test means.

	$\Pr[b_t=0]$	$\Pr[b_t,b_{t+1}=0]$	$\Pr[b_t,\cdots,b_{t+2}=0]$	$\Pr[b_t,\cdots,b_{t+3}=0]$
Sample Mean	$\bar{x}_f=0.5707$	$\bar{x}_f=0.3082$	$\bar{x}_f=0.1718$	$\bar{x}_f=0.0948$
Test Mean	$\bar{x}_f=0.5625$	$\bar{x}_f=0.3125$	$\bar{x}_f=0.1719$	$\bar{x}_f=0.0938$
T-Statistics	$T_f=-0.706$	$T_f=0.4928$	$T_f=0.0107$	$T_f=-0.167$
TINV	$T=1.66$	$T=1.66$	$T=1.66$	$T=1.66$
P-Value	$T_{\text{tail}}=0.482$	$T_{\text{tail}}=0.623$	$T_{\text{tail}}=0.992$	$T_{\text{tail}}=0.868$

To be specific, we generate 200 files that include only ASCII characters selected uniformly and independently at random regardless of the position of the characters. Then, we apply Step 2 and 3 of Section 3.3 to obtain 400,000 scores for each of $\ell=0,1,2,3$. We call these scores test scores to distinguish them from the sample scores generated in Section 3.3.

To verify this statistically, we apply two-sample F-test to the sample scores and the test scores. Consequently, the null hypothesis to use should be that the population variances underlying the sample scores are identical to those underlying the test scores.

Table 2 shows the result of the test, where σ_f^2 and $\sigma_t^2, \bar{x}_f$ and $\bar{x}_t$ are the variances and the means of the sample scores and the test scores, respectively. According to **Table 2**, we see that the variance σ_f^2 is greater than σ_t^2, while the means x_f and x_t are very close to each other. When investigating the *P*-values presented in the table with the significance level 0.05, we see that the null hypothesis should be rejected for all cases of $\ell=0,1,2,3$. Thus, when we apply URDA to ASCII files downloaded from the Internet with the fixed fragment length seven bits, the resulting archived files will not be explained by the mathematical model that is a basis of Equation (1). In other words, we cannot prove that the archived files generated by URDA with the aforementioned setting are secure against the bit pattern analysis, and we cannot recommend use of URDA with this setting.

3.5. URDA with the Setting of the Fragment Length = Variable

As seen in the above, using URDA with the fixed fragment length cannot be recommended from the security point of view. In this clause, we investigate the security of URDA with a different setting, that is, we assume that the fragment length is variable, and is selected at random per fragment in the range of 1 to 32 bits.

For the experiment, we use the same ASCII files stated in Section 3.3. The difference consists in Step 2, and we run URDA with the setting of the fragment length = variable.

In the same way as Section 3.3, we obtain 400,000 sample scores for each of $\ell=0,1,2,3$, and plot them in **Figure 5**.

At a glance, the means of the sample scores are identical to their theoretically expected values of $\Pr[b_t=\cdots=b_{t-\ell}=0]=\frac{l+9}{2^{l+4}}$, and the variances have become smaller compared with the case of the fixed fragment length. In the following, we verify these observations by means of statistics.

3.5.1. Examining the Means

In the same way as the previous subsection, we apply the one-sample *t*-test to verify that the population means of the sample scores are identical to the theoretical expectation.

Table 3 shows that the *P*-values obtained are large (the minimum is 0.188), and we cannot reject the null hypotheses with a small significance level, for example, 0.05. Hence the population means of the target probabilities are identical to the values given by $\frac{l+9}{2^{l+4}}$ for $\ell=0,1,2,3$.

3.5.2. Examining the Variances

As shown in **Table 4**, the variances of the sample scores are close to those of the test scores, which are the scores generated in Section 3.3 using the synthetic ASCII files. If this is true, this will be a clear contrast with the former case where we ran URDA with the fixed fragment length. In the following, we verify that the population variances of the sample scores and the test scores are identical to each other by means of the two-sample *F*-test.

Table 4 shows that the *P*-values obtained are large (the smallest is 0.1704), and we cannot reject the null hypotheses with a small significance level, for example, 0.05.

As a consequence, we can conclude that the population variances that underlie the sample scores and the test scores are identical to each other.

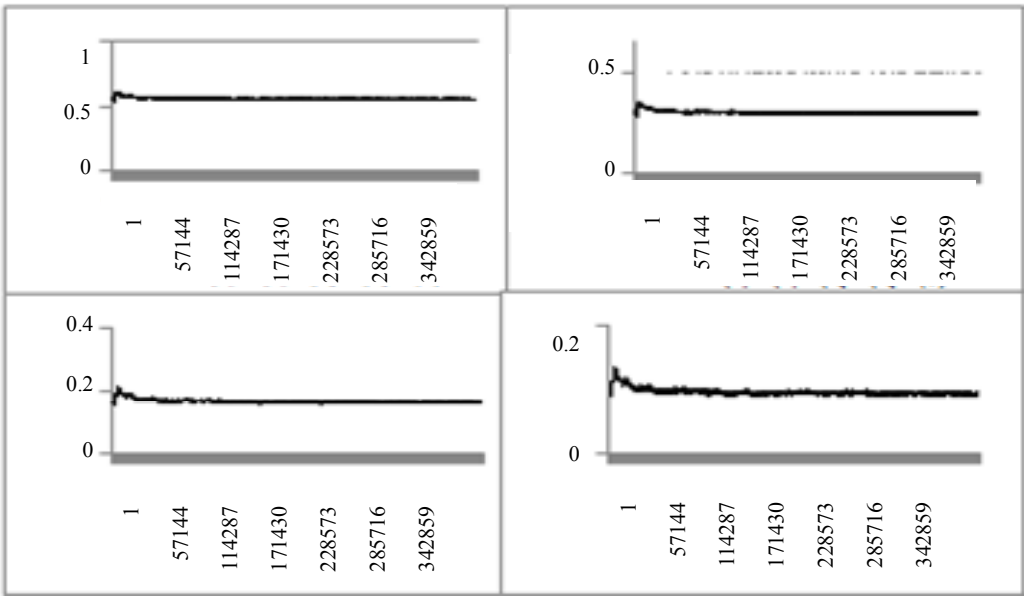

Figure 5. Variable-fragment-length Pr[$b_t = \cdots = b_{t-l} = 0$] (l = 0,1,2,3, t = 1, ⋯ , 400,000).

Table 2. Comparing of fixed variances and test variances.

	$\Pr[b_t = 0]$	$\Pr[b_t, b_{t+1} = 0]$	$\Pr[b_t, \cdots, b_{t+2} = 0]$	$\Pr[b_t, \cdots, b_{t+3} = 0]$
Sample Variance	$\sigma_f^2 = 0.0135$	$\sigma_f^2 = 0.0078$	$\sigma_f^2 = 0.0078$	$\sigma_f^2 = 0.0036$
Test Variance	$\sigma_f^2 = 0.0036$	$\sigma_t^2 = 0.0045$	$\sigma_t^2 = 0.0019$	$\sigma_t^2 = 0.0014$
Sample Mean	$\bar{x}_f = 0.5707$	$\bar{x}_f = 0.3082$	$\bar{x}_f = 0.1718$	$\bar{x}_f = 0.0948$
Test Mean	$\bar{x}_t = 0.5617$	$\bar{x}_t = 0.3103$	$\bar{x}_t = 0.1721$	$\bar{x}_t = 0.0923$
F-Value	1.53	1.72	2.64	2.61
Critical Value	$O_{\text{tail}} = 1.39$ $T_{\text{tail}} = 1.49$	$O_{\text{tail}} = 1.39$ $T_{\text{tail}} = 1.49$	$O_{\text{tail}} = 1.39$ $T_{\text{tail}} = 1.49$	$O_{\text{tail}} = 1.39$ $T_{\text{tail}} = 1.49$
P-Value	$T_{\text{tail}} = 0.036$	$T_{\text{tail}} = 0.007$	$T_{\text{tail}} = 0.000002$	$T_{\text{tail}} = 0.000004$

Table 3. Comparing of variable means and test means.

	$\Pr[b_t = 0]$	$\Pr[b_t, b_{t+1} = 0]$	$\Pr[b_t, \cdots, b_{t+2} = 0]$	$\Pr[b_t, \cdots, b_{t+3} = 0]$
Test Mean	$\bar{x}_t = 0.5625$	$\bar{x}_t = 0.3125$	$\bar{x}_t = 0.1719$	$\bar{x}_t = 0.0938$
Sample Mean	$\bar{x}_r = 0.5511$	$\bar{x}_r = 0.3031$	$\bar{x}_r = 0.1712$	$\bar{x}_r = 0.0938$
TINV	$T = 1.66$	$T = 1.66$	$T = 1.66$	$T = 1.66$
T-Statistics	$T_r = 1.067$	$T_r = 1.3261$	$T_r = 0.1437$	$T_r = -0.021$
P-Value	$T_{\text{tail}} = 0.288$	$T_{\text{tail}} = 0.188$	$T_{\text{tail}} = 0.886$	$T_{\text{tail}} = 0.983$

Table 4. Comparing of variable variances and test variances.

	$\Pr[b_t=0]$	$\Pr[b_t,b_{t+1}=0]$	$\Pr[b_t,\cdots,b_{t+2}=0]$	$\Pr[b_t,\cdots,b_{t+3}=0]$
Sample Variances	$\sigma_r^2=0.0115$	$\sigma_r^2=0.0051$	$\sigma_r^2=0.0025$	$\sigma_r^2=0.0015$
Test Variances	$\sigma_t^2=0.0088$	$\sigma_t^2=0.0045$	$\sigma_t^2=0.0019$	$\sigma_t^2=0.0014$
Sample Mean	$\bar{x}_r=0.5511$	$\bar{x}_r=0.3031$	$\bar{x}_r=0.1712$	$\bar{x}_r=0.0938$
Test Mean	$\bar{x}_p=0.5617$	$\bar{x}_p=0.3103$	$\bar{x}_p=0.1721$	$\bar{x}_p=0.0923$
F-Value	0.7694	0.89	0.7583	0.9458
Critical Value	$O_{\text{tail}}=1.39$ $T_{\text{tail}}=1.49$	$O_{\text{tail}}=1.39$ $T_{\text{tail}}=1.49$	$O_{\text{tail}}=1.39$ $T_{\text{tail}}=1.49$	$O_{\text{tail}}=1.39$ $T_{\text{tail}}=1.49$
P-Value	$T_{\text{tail}}=0.194$	$T_{\text{tail}}=0.549$	$T_{\text{tail}}=0.1704$	$T_{\text{tail}}=0.7823$

3.6. Summary of This Experiments

In our previous work, assuming that source files include ASCII characters selected uniformly and independently at random per character, we have mathematically proved that an archived file consisting of seven-bit-long fragments shows uniformly constant distributions with respect to the probabilities that particular bit patterns appear. In other words, we can view the archived file as secure against the bit pattern analysis. On the other hand, the distributions of characters that appear in files of the real world are certainly biased, and hence, it was not certain that the same conclusion holds true when applying URDA to files of the real world with the same setting. To investigate this problem, we first compared the following two scenarios in terms of the means and variances of the probability distribution of some particular bit patterns.

- Apply URDA to files taken from the Internet with the setting of seven-bit-long fragments.
- Apply URDA to synthetic files generated so that every ASCII character appears with the same probability regardless of the position in the files.

As we expected, the means of the probability are identical to their theoretical expectation of $\frac{l+9}{2^{l+4}}$ for $\ell=0,1,2,3$, but the variances for the first scenario are significantly larger than those for the second scenario. Therefore, we cannot deny the possibility that there exist some clever attacks that take advantage of this difference in the variance.

Hence, we ran the same experiments after modifying the first scenario. In the new scenario, the length of each fragment is not fixed, but is determined at random between 1 bit to 32 bits per fragment. The results of the comparison is surprising, we cannot detect statistically significant differences in either the means or the variances between the scores for the downloaded and randomly fragmented ASCII files and those for the synthetic ASCII files. In other words, we cannot statistically distinguish between the archived files generated from the downloaded files and the synthetic files.

Thus, we cannot conclude that use of URDA with the fixed fragment length is secure, while use of URDA with the randomly variable fragment length is secure against bit pattern analysis.

4. Conclusion and Future Work

We investigated whether the formula

$$P[b_t=\cdots=b_{t-l}=0]\approx\frac{l+9}{2^{l+4}}$$

applies to real ASCII text files that include biases in the distributions of characters. Applying URDA to downloaded text files by fragmenting the files into seven bit long, we found that the resulting archived files might be vulnerable to bit pattern analysis attacks. Thus, in the same way as the previous experiment, we ran another experiment by modifying URDA in the way how to select fragment lengths, that is, the length of each fragment is determined at random per fragment between 1 bit to 32 bits, and showed this modified URDA with the randomly variable fragment length is secure against the bit pattern analysis.

As the next step, we will investigate the security of URDA when applied to text files that include a variety of character sets and their corresponding code units. We will specify the significance of biased bits in these code units, which includes their actual representation in storages and how it affects security issues in plain text archiving. Consequently we need to testify, by stochastically determining the size *b* per fragment, whether we can obtain similar conclusion for various kinds of source

files with various bit distributions. In addition, we also investigate some sorts of file existed in the real world that are said close to random files like video, image and compressed files etc.

REFERENCES

[1] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39612

[2] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz and D. A. Patterson, "Aid: High-performance, Reliable Secondary Storage," *ACM Computing Surveys*, Vol. 26, No. 2, 1994, pp. 145-185.

[3] A. Shamir, "How to Share a Secret," *Communication of ACM*, Vol. 22, No. 11, 1979, pp. 612-613.

[4] G. Blakley, "Safeguarding Cryptographic Keys," 1979 *Proceedings of the National Computer Conference*, New York, 4-7 July 1979, p. 313.

[5] M. O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance," *Journal of the ACM*, Vol. 36, No. 2, 1989, pp. 335-348. doi:10.1145/62044.62050

[6] L. Bai, "A Strong Ramp Secret Sharing Scheme Using Matrix Projection," *Proceedings of the* 2006 *International Symposium on a World of Wireless*, *Mobile and Multimedia Networks*, New York, 26-29 July 2006, pp. 652-656.

[7] L. Bai and X. K. Zou, "A Proactive Secret Sharing Scheme in Matrix Projection Method," *International Journal of Security and Networks*, Vol. 4, No. 4, 2009, pp. 201-209.

[8] C. Blundo, "Alfredo de Santis and Ugo Vaccaro, Efficient Sharing of Many Secrets," Springer Verlag, Berlin, 1993.

[9] J. M. He and E. Dawson, "Multistage Secret Sharing Based on One-Way Function," *Electronic Letters*, Vol. 30, No. 19, 1994, pp. 1591-1592. doi:10.1049/el:19941076

[10] K. Wang, X. K. Zou and Y. Sui, "A Multiple Secret Sharing Scheme Based on Matrix Projection," *Proceedings of the* 33*rd Annual IEEE International Computer Software and Applications Conference*, Seattle, 20-24 July 2009, pp. 400-405.

[11] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon and J. Kubiatowicz, "Maintenance-Free Global Data Storage," *IEEE Internet Computing*, Vol. 5, No. 5, 2001, pp. 40-49. doi:10.1109/4236.957894

[12] A. Rowstron and P. Druschel, "Storage Management and Caching in PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility," *Proceeding SOSP*'01 *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, Banff, 21-24 October 2001, pp. 188-201.

[13] A. Tallat, K. Shin, H. Lee and H. Yasuda, "Some Remarkable Property of the Uniformly Random Distributed Archive Scheme," *Advances in Information Sciences and Service Sciences*, Vol. 4, No. 11, 2012, pp. 114-124.

[14] Cisco System, "Storage Networking 101," Cisco System, San Jose, 2001.

[15] International Business Machines Corporation, "Introduction to Storage Area Networks," 2012. http://www.redbooks.ibm.com/

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A scalable Content-Addressable Network", *SIGCOMM* '01 *Proceedings of the* 2001 *Conference on Applications*, *Technologies*, *Architectures*, *and Protocols for Computer Communications*, 2001, pp. 161-172.

[17] International Organization for Standardization, "ISO/IEC 7816 Series: Identification Cards-Integrated Circuits(s) Cards with Contacts," International Organization for Standardization, Geneva, 1999.

[18] International Organization for Standardization, "ISO/IEC 18092-3: Information Technology-Telecomunications and Information Exchange between Systems-Near Field Communication-Interface and Protocol (NFCIP-1)," International Organization for Standardization, Geneva, 2004.

[19] IBM Corporation, "Infrared Data Association: Serial infrared Link Access Protocol (IrLAP) Version.1.1," IBM Corporation, New York, 1996.

