Vol.2011-SPT-2 No.1
2011/12/26

Some remarkable property of the uniformly random distributed archive scheme

Ahmed Tallat, Tokyo Denki University
Hiroshi Yasuda, Tokyo Denki University
Kilho Shin, University of Hyogo
Hwaseon Lee, Korea University

Abstract— **this paper proposes distributed archive scheme that brings ubiquitous environment into real, with threshold scheme of both (*k*, *n*) robustness and (*k*, *n*) security. In the proposed scheme, we demonstrated fast and practically secure distributed archive scheme, in which each piece of source file is archived into randomly chosen (*n*-*k*+*1*) storages out of *n* storage. Later, to be able to download from *k* storages is enough to recover original file, but even (*k*-*1*) storages are attacked, the stolen data cannot cover total amount of the file. The proposing scheme is robust enough to stand against unsteadiness of the Internet and protects the secrecy of data without relying on cryptographic technics.**

1. INTRODUCTION

Digitalizing records provide easier data management task for the rapid increase amount of data, which in turn calls for the need toward storage devices to maintain them properly. However, to guarantee the availability, accessibility and confidentiality of the data stored in storage medium is not easy task. Because every system is vulnerable to certain damages caused by both physical and technical attacks, and thus these data can be lost, delayed or even stolen, which would cost unprecedented troubles. Fortunately JBOD and RAID [1] were created to address these problems, equipped with some technical schemes that are deployed for providing security and redundancy to enable data recover in case of disaster or system failure.

However, the widespread use of the Internet affecting the way information being accessed, stored and shared posed strong demand to the scale of flexibility of storages for these ever-increasing data. Because conventional way of managing data, that is, every server has own individual storage inside its cabinet, is no longer able to meet the need that the Internet brings for sharing huge amount of data in terms of variability and amount. That is where network storage systems like Sustor [2], NAS (Network Attached Storage)[3] and SAN (Storage Area Networks) [4] come to address these problems providing new way of archiving files.

Yet, the Internet is unstable. Applying ubiquitous access into the Internet, there are risks of retrieving the archived data securely, because it is common that servers are down, accesses are denied, and transmitted data is corrupted etc. Thus, the need for robust and secure system for protecting our sensitive data is crucial concern—be it individual or any organization. To address the instability of the Internet, we propose some remarkable properties of the uniformly random distributed archive (URDA) scheme.

URDA consists of smartcard and host that can be client software or server. The role of smartcard is to generate random distribution keys, whereas host does archiving following the indication by the random distribution keys. There are both archiving phase and retrieving phase in URDA, and simple descriptions of them are stated as follows.

During the archiving phase, host fragments archiving file into *N* pieces, and requests smartcard to send random distribution keys that indicates (*n*-*k*+*1*) destination files out of *n* files (*k*≤*n*). Following the indication of the sequence of the distribution keys, the host takes current fragment from the source file and attaches a copy of the fragment to each of the (*n*-*k*+*1*) destination files indexed by current distribution key. This process repeats itself until whole fragments are attached to the whole predetermined *n* destination files. Finally, each destination file is send to independent *n* storages individually.

During the retrieving phase, users need to download data fragments from at least *k* storages out of *n* storages and the data fragments included in these storages are enough to cover whole fragments of original file. However, even the data fragments in (*k*-*1*) storages are stolen, the theft is unable to recover the original file since the stolen fragments do not cover total fragments necessary to recover original source file.

Following sections are organized as: section 2 includes related works. Requirements and algorithm are given in section 3. The proof of (*k*, *n*) robustness and (*k*, *n*) security is described in section 4 with detailed analyses and we conclude the paper in section 5 with future plan.

2. RELATED WORK

As related work following subsections describe secret sharing scheme, RAID and sustor.

2.1 SECRET SHARING SCHEME

Cryptography is common technology for protecting security in everyday life and key is essential to make it happen. Now the question is how about the security of the key, which can be lost, destroyed and forgotten. Making duplicated copies of the key can prevent these attacks, but the key would be exposed for more chances to be stolen. That is where (*k*, *n*) threshold secret sharing scheme was born to solve such dilemma.

Secret Sharing Scheme is way of creating multiple pieces from a secret and collecting certain number of the pieces can recover the secret, but less pieces than the certain number cannot reveal any information about the original secret. Adi Shamir created (*k*, *n*) threshold secret sharing scheme [5], which means *k* shadows out of *n* created from a secret is sufficient to recover the secret, however, any (*k*-*1*) shadows or less reveals no information about the secret. Actually there are many ways of showing the secret sharing protocols, following we utilize polynomial and Lagrange to demonstrate how it works practically.

Polynomial is sum of terms that consists of variables raised by whole number exponents indicating its degree and constant.

Usually it is written in decreasing order of exponents. To apply Polynomial to show secret sharing protocol, we know that points on a curve can verify polynomial, that is, a polynomial with certain degree can be found out by figuring out the value of certain points that it goes through—point's number equals degree number plus one. For example, if the threshold in Shamir`s secret sharing scheme is k, then k-1 degree polynomial that has k-1 terms is used for the calculation and the last term without variable corresponds to the secret written as $f(x) = f_{k-1}x^{k-1} + \cdots + f_2x^2 + f_1x + f_0$, [$(f_0) = S$]. Once choosing random numbers for other coefficients, we can compute shadows from the calculation of corresponding f(x) for each x such as shadows are pairs of $(x_i, f(x_i))$ ($i \in [1, n]$). Thus, none of the pairs reveals any information about the f_0, if we destroy the randomly selected numbers for the coefficients.

On the other hand, to recover the secret [$(f_0) = S$], we use Lagrange Interpolation to compute the polynomial back from any k shadows, that is, $(x_1, f(x_1))$ $(x_2, f(x_2))$ … $(x_i, f(x_i))$ ($i \in [1, k]$). We can calculate $f(x) = \sum_{i=1,\dots,k} f(x_i) * l_i(x)$, where $l_i(x) = \prod_{i=1}^{k} \frac{x - x_i}{x_j - x_i}$ $(j = 1, \dots, k)$.

Blakley [6] also presented (*k, n*) threshold secret sharing scheme independently. Since then the approach has been widely studied and applied on key management, access control, storage of data, visual cryptography etc.

(*k, n*) threshold scheme is mature and well-studied protocol. Although we mentioned that any k-1 shadows cannot recover original secret, but we must differentiate the difference between leaking no information about secret and recovering secret. The former is called perfect secret sharing because no information is revealed from k-1 or fewer shadows regardless of time and power that attackers may have, and it is also said that the size shadow is same as the secret size. In other word, if the size is smaller than the secret size, then it is not called perfect secret sharing. Hence, there is another type of threshold scheme called ramp secret sharing [7], in which shadow size and secret are proportional. Without ramp secret sharing, it would be challenging to apply the protocol to large files and therefore the ramp secret sharing scheme meets the need of reducing shadow size at expense of leaking secrecy at some extend. Taking advantages of matrix projection Li[8][9] proposed strong ramp secret sharing scheme, in which secrets represent elements in a square matrix providing significant reduction of shadow size and protection of multiple secrets.

In this paper we further take advantage (*k, n*) threshold scheme putting its potential usability into distributed storage system in ubiquitous environment, in which anyone can archive and retrieve any data from anywhere anytime. Traditional threshold secret sharing schemes (eg. Matrix projection, polynomial etc) may not be adequate to apply distributed archive system due to its heavy calculation power. Thus, the paper comes from the need to meet practical execution speed and thus there is tradeoff between security and execution speed.

2.2 RAID

In order to deal with a large amount of data, the trend of computer development called for the need of high capacity, performance, redundancy and scalability of hard disks. For example, with traditional approach, once a disk dies, neither upload nor download works. Thus, we need another disk that carries on the failed disk`s task. This can be accomplished by establishing identical disks being updated as virtualized single disk. Therefore, getting together a bunch of disks, it is possible to reach desired functionalities mentioned above. Following describes how the needs are met by stating variety of RAID (*Redundant Array of Inexpensive Disks*) forms.

- **Concatenation**

This is simple RAID; getting together all the disks available and creating a single virtualized one—the capacity of the virtualized one equals adding each one participated in. But with this type, disks are no use unless the previous one is filled up. Because data is stored in next disk only after the first one is full. With the simple RAID, the role of having 100TB equals to 10GB, if the dealing data is less than 10GB.

- **Striping—RAID-0**

RAID-0 is same as simple RAID in terms of virtualizing, that is, the former also get a bunch of disks together to create a single virtualized one. However, unlike the simple RAID, it using RAID-controller stripes data across all the disks simultaneously and thus it increases both writing and reading speeds. We can understand the speed improvement performance from an example, where 1 GB is stored to the RAID-0 and simple RAID. Let us assume that the data is striped by 4 bits and that it stores each 4 bits in a disk one at a time. With RAID-0, each unique 4 bits is stored in multiple disks simultaneously one at a time, whereas only 4 bits data stored a simple RAID disk one at a time. Thus the speed of RAID-0 is n (number of disks) times faster than the simple RAID. But the problem with RAID-0 is that it provides no redundancy since failure of any one of the disks results in total loss of the data stored into the single virtualized disk.

- **Mirroring—RAID-1**

RAID-1 aimed at providing redundancy by establishing duplicated disks, each of which stores same data. The level of the redundancy is determined by the number of disk copies, thus any of the non-crushed disks carries on the task of a crushed one to continuing of its tasks without corruption. Further, mirroring is possible for both simple RAID and RAID-0; however, such redundancy comes at high expense of coping disks.

- **Stripping plus Mirroring—RAID-0+1**

If the mirroring technique applied to simple RAID, then we will need another bunch of duplicated disks to provide redundancy. If it applied to RAID-0, which is called RAID-0+1, then we would need same amount of disks, on which data is striped exactly same way as the striped RAID-0 targeted for protection.

- **Striping with Parity (RAID-5)**

At least three disks are needed to establish RAID-5, in which, unlike RAID-0, RAID-controller not only stripes data across disks, but also creates parity spread across disks using XOR function. For example, first striped data stored in first disk,

next striped data stored in second disk, third disk stores parity data calculated from the first two striped data using XOR function. However, there is no dedicated disk that stores parity data, rather each disk has own turn to store pieces of parity data in turn. The multiple simultaneous read capacities contribute to performance speed, whereas parity provides redundancy skills. However, calculating parity consumes spaces and CPU power for the calculation.

By the way, RAID-2 and RAID-3 are pretty much like hybrid of mirroring and striping, but almost not used anymore. RAID-4 and RAID-6 is also striping and calculating parity, but it is, unlike RAID-5, the parity data of RAID-4 is stored in a dedicated disk, whereas RAID-6 does two parity striping so as to stand against two disk failures.

From the analyses above, we can see that RAID provides some extension of redundancy and space capacity, but *k* is confined within 1, n-1 and n-2 for applying to (*k, n*) threshold scheme. However, such limitation poses strong restrictions on implementation of redundancy into distributed storage network system.

2.3 SUSTOR

Sustor consists of two type nodes (source node and storage node) and two type networks (Data access overlay network (DAO) and Data disseminate overlay network (DDO)). Storage node has functions of storing and reading data, whereas Source node, other than possessing the function of storage node, has functions of assigning a number to each piece of data and sending the data to the storage nodes. DAO network is used for connecting neighbor nodes, calculating data ID stored at each node and assuring the accessibility of these data for each node. DDO network is used to disseminate the data by connecting the nodes holding replica of same pieces of data based on the data ID determined in DAO network.

Initialization of a new Sustor: First, each storage node passes a ticket with its network address to source node for validating its authenticity and the source node responds with node list and shortcut path to all storage node in its list. Next, on receipt of data, each Storage node, based on the number of nodes and its own ID, establishes DAO and DDO network through TCP connection. Last, each node monitors if its neighbors are active by periodic heartbeat messages.

- **Data access overlay network**

DAO is used by nodes to access data that the nodes hold. Which pieces of data should be stored on which node is determined by the $i_n + L - (2k+1)j \bmod L \ \left(0 \le j \le \left\lfloor \frac{L}{2k} + 1 \right\rfloor\right)$ formula, where L is number of nodes and k is range of neighborhood. E.g. a piece of data with $i_d < L$ is stored in the same place with the $i_d \bmod L$. Therefore, the formula enables each node to determine which piece of data the node should store as well as the locations of pieces of data in the neighboring nodes have, and thus any node can reach a desired piece of data stored in a node determined by the equation as long as the links between the nodes are active.

- **Data disseminate overlay network**

Every node establishes TCP connections to the nearest nodes holding same piece of data, which constitutes **DDO** overlay network to disseminate pieces of data. If there is shortcut link in DAO network, then it is used by nodes to find their nearest node too. A node determines its neighboring nodes that hold same piece of data in DDO network by propagating small messages and sending them in both directions in the DAO ring, and other nodes forward the received message in the same direction. Once the neighbors that hold the same piece of data are figured out, each node in DDO waits to forward the received messages to other neighbors.

- **Reconfiguring Overlay Network on Failure**

If node failure occurs in DDO, the reconfiguring processes described above can be taken. If node failure occurs in DAO, any node who desires the pieces of data that the failed node hold propagates messages and sends them to other nodes. Once the substitute node holding the replica of the failed node is found out, the sender node connects to the substitute node so that the accessibility of all pieces of data is assured.

Sustor aimed at improving tolerance against large scale of disaster by enabling providers jointly to share data. However, with URDA, anyone archiving data retrieves own files without sharing.

3. UNIFORMLY RANDOM DISTRIBUTED ARCHIVE SCHEME

First we will describe three requirements that the proposed scheme aim at satisfying, then we introduce our algorithm with more details in both archive phase and retrieve phase.

3.1 REQUIREMENTS

This paper`s target is to meet following three requirements.

- **Ubiquitous access:**

Ubiquitous access indicates environment, in which anyone can archive and retrieve any information from anywhere anytime, at the same time the data requirement towards security and availability should be satisfied. We implement smartcard to index how the fragments of a source file are distributed and retrieved in URDA. Besides, smartcard also plays the role of access control function, which means the same smartcard used for archiving is essential for retrieving as well.

- **(k, n) robustness**

Archived files mostly are stored in outsiders` storages in network clouding system, out of the file owner`s control. Therefore, there is no guarantee that users can retrieve their archived files anytime, anywhere due to the instability of the Internet such as denying of access, down of server, damages of files etc. The proposed scheme addresses these instability

problems of the Internet by satisfying (*k*, *n*) robustness requirement.

Distributed archive system is said to be (*k*, *n*) robustness if original data can be recovered by retrieving archived data from *k* storages out of *n*. On the other hand, even (*n*-*k*) storages are out of access, users still can recover original source file, due to the (*n*, *k*) robustness that the URDA provides.

- **(k, n) security**

Varieties of cryptographic schemes are commonly employed to keep the data in secure and key is essential to make it work. However, cryptography cannot prevent the corruption of data during transmission and the theft from stealing stored data. Neither is the confidentiality of data guaranteed if the key is tempered. The proposed scheme addresses these problems by satisfying (*k*, *n*) security without relying on cryptography.

Distributed archive system is said to be (*k*, *n*) secure if retrieving archived data from *k-1* storages out of *n* storages cannot cover the total amount of original data, which means even if attackers are capable to steal archived data fragments from (*k-1*) storages, they still unable able to reconstruct original source file.

3.2 ALGORITHEM

The proposed scheme of distributed archive system consists of archiving phase and retrieving phase, and smartcard and host communicate to each other in the two phases during archiving and retrieving process. The main role of smartcard is to generate random distribution keys, each of which determines (*n*-*k*+*1*) destination files out of *n* destination files in order to attach a copy of current fragment to each of the (*n*-*k*+*1*) files. Accordingly the key space equals $\boldsymbol{nC_{n-k+1}} = \frac{\mathbf{n!}}{\mathbf{(n-k+1)!(k-1)!}}$. On the other hand, host fragments archive file into *N* pieces with same length *b* and prepare *n* destination files. Once receiving the distribution keys from smartcard, the host takes away current fragment from source file and attaches it to (*n*-*k*+*1*) destination files indicated by the key. This process is repeated until all the fragments of source file are attached and then the *n* destination files are sent to *n* independent storages. Simple image of the algorithm is shown in Fig 1.

- **Archiving phase**

Smartcard contains both random distribution key generating module (RNG_k) and seed generating module (RNG_s). Duration of archiving, receiving the file name to be archived of, RNG_s generates seed to intrigue RNG_k for generating distribution key, and the seed is also stored with received file name together as pair to be used during retrieving process. Then the smartcard keeps sending the resulted distribution key to host until each fragment of source file is distributed into *n* destination files.

On the other hand, host cut the file into pieces, each of which is same length of *b*, and prepare *n* destination files. Once receiving the distribution key from smartcard, the host takes away current fragment from source file and attach it to (*n*-*k*+*1*) destination files out of *n* destination files following the indication of the distribution key. The process is repeated until all the fragments are distributed fully to these destination files. Then the *n* destination files are independently sent to *n* storages in network clouding systems.

- **Retrieving phase**

Retrieving phase is another way of doing the communication between smartcard and host. First the user needs to download destination files from at least *k* storages out of *n* storages. As alternative way, first user downloads a destination file from storage, and compensates lost fragments from other available storages. In this way, users can decrease the total amount of downloading data, which in turn helps to save resources like memory space, CPU computational power and communication bottleneck etc.

Once receiving the archived file name from host, the smartcard retrieves the seed paired with the file name. Input the seed into RNG_k to generate sequence of distribution key, each of which is used to indicate (*n*-*k*+*1*) destination files that includes current fragment.

On the other hand, receiving the distribution key from smartcard, the host can evaluate the corresponding position of each fragment necessary to reconstruct the original source file.

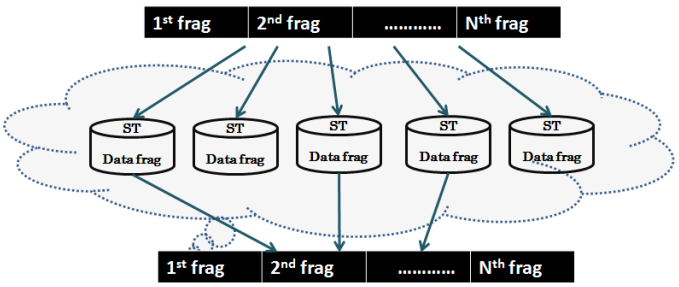

Fig.1. uniformly random distributed archive (n=5, k=3)

4. PROOF

In this section we will analysis detailed proof of the (*k*, *n*) robustness and (*k*, *n*) security that URDA provides.

4.1. (*k*, *n*) ROBUSTNESS

In the proposed algorithm, each fragment is included in (*n*-*k*+*1*) storages, and the number of storages that does not include a specific fragment is (*k-1*). Therefore, the proposed scheme meets (*k*, *n*) robustness requirement.

Assuming that users retrieve data from r storages ($n \geq r$) and that p(r; n, k) denotes the probability of a specific original data

fragment that is not included in the r storages. Then, this holds;

$$p(r;n,k)=\prod_{i=1}^{r}\frac{k-i}{n+1-i}$$

Proof: when a storage randomly selected, the probability of a specific original data fragment that is not included in the storage equals $\frac{k-1}{n}$. The probability of next storage that does not contain the fragment is $\frac{k}{n-1}$ and so on. Hence, the probability of the specific fragment that is not included in r storages equals $\frac{k-1}{n}*\frac{k-2}{n-1}*\ldots*\frac{k-r}{n+1-r}$. Thus we can conclude that;

$$p(r;n,k)=\frac{k-1}{n}*\frac{k-2}{n-1}*\ldots*\frac{k-r}{n+1-r}=\prod_{i=1}^{r}\frac{k-i}{n+1-i}$$

4.2. SECURITY

Our proposed scheme does not support theoretical information security of (*k*, *n*) threshold scheme; rather it emphasizes the practicability of archiving system that provides time efficiency of highly execution speed and practical privacy. Following we demonstrate the security aspect of the scheme.

No safety measurement is necessarily unless otherwise it would cause certain loss due to any possible attack. Thus, safety measures are commonly shaped by the seriousness of such loss. Following we investigate on the strength of security of our algorithm against possible attacks.

1. Security against specifying all the fragments of original file.

The probability of a specific original file fragment included in a storage out of *n* storages is $\frac{n-k+1}{n}$, second storage is $\frac{n-k}{n-1}$ and so on. Hence, the probability of a specific fragment included in r storages is $\frac{n-k+1}{n}*\frac{n-k}{n-1}*\ldots*\frac{n-k-r}{n-r-1}$. In other words, we can also reach it by;

$$1-p(r;n,k)=\frac{n-k+1}{n}*\frac{n-k}{n-1}*\ldots*\frac{n-k-r}{n-r-1}=\prod_{i=1}^{r}\frac{n-k-i}{n-i-1}$$

Therefore, if r storages are attacked, the probability that the attacker specifies all the fragments of original file is

$$(1-p(r;n,k))^{N}=\left(\frac{n-k+1}{n}*\frac{n-k}{n-1}*\ldots*\frac{n-k-r}{n-r-1}\right)^{N}=\left(\prod_{i=1}^{r}\frac{n-k-i}{n-i-1}\right)^{N}$$

, where *N* is total fragment number of original file. When $r\geq k$, $(1-p(r;n,k))^{N}=1$, which means the attacker can identify all the fragments. When r<k, it approaches 0, as N gets larger enough.

2. Security against specifying all the fragments of revealed sequence

Fallowing assumptions are taken for granted:

1) The order of fragments in revealed sequence is identical to original file.

2) The length of each fragment that is fixed is not secret.

3) Fragment content is independent from its position in original file.

Let us assume that attacker obtains a destination file from storage. Then attacker fragment the file based on predetermined knowledge of fragment length and knows that the order of the fragments is same to original file. Since the content of each fragment reveals no information about its particular position in source file, the attacker has to rely on probability guess.

There are *n* independent storages, and the probability of a particular fragment included in the revealed sequence is $\frac{n-k+1}{n}$. On the other hand, the probability of the fragment excluded from the revealed sequence is $\frac{k-1}{n}$. Additionally, assuming that a revealed sequence has *m* fragments (*m*<*N*), there are *N*-*m* fragments excluded from the revealed sequence. Hence, the probability of identifying the original positions of all the fragments in the revealed sequence is

$$p(1;n,k)^{N-m}\,(1-p(1;n,k))^{m}=\left(\frac{k-1}{n}\right)^{N-m}\left(\frac{n-k+1}{n}\right)^{m}=\frac{(k-1)^{N-m}\,(n-k+1)^{m}}{n^{N}}$$

While r<k, the probability approaches to 0 as the *N* get increased.

3. Security against identifying a single fragment

It makes sense that a single fragment might have certain significance to attacker and the attacker might be interested of finding its position in revealed sequence. Fallowing we find out of such likelihood.

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

Since *m* fragments are randomly picked up from *N* fragments, we can write that sample space equals ${}_N\boldsymbol{C}_m$. Furthermore, as we stated that the order of fragment in source file is preserved even in revealed sequence, (*j-1*) fragments in the revealed sequence must be selected from (*a-1*) fragments in source file, which means (*m-j*) fragments in revealed sequence must be selected from (*N-a*) fragments in source file. Thus the numerator equals ${}_{a-1}\boldsymbol{C}_{j-1} \cdot {}_{N-a}\boldsymbol{C}_{m-j}$. As the result the probability of identifying a single fragment equals

$$\mathbf{Succ}(\boldsymbol{a},\boldsymbol{j};\boldsymbol{N},\boldsymbol{m}) = \frac{{}_{a-1}\boldsymbol{C}_{j-1} \cdot {}_{N-a}\boldsymbol{C}_{m-j}}{{}_N\boldsymbol{C}_m}$$

Since *a* and *j* may not necessarily be fixed, therefore following properties should be investigated.

1) When *a* and *j* move, what is maximum value of $\mathbf{Succ}(\boldsymbol{a},\boldsymbol{j};\boldsymbol{N},\boldsymbol{m})$?

When *a=1*, *j=1* or *a=N*, *j=m*, it reaches its maximum value of *m/N* as calculated below, which means the first fragment or last fragment of source file appear at the first or last position of revealed sequence.

$$\mathbf{Succ}(\mathbf{1},\mathbf{1};\boldsymbol{N},\boldsymbol{m}) = \mathbf{Succ}(\boldsymbol{N},\boldsymbol{m};\boldsymbol{N},\boldsymbol{m}) = \frac{{}_{N-1}\boldsymbol{C}_{m-1}}{{}_N\boldsymbol{C}_m} = \frac{(N-1)!}{(N-m)!(m-1)!} * \frac{(N-m)!m!}{N!} = \frac{m}{N}$$

Following we prove $\sum_{j=1,\dots,m} \mathbf{Succ}(\boldsymbol{a},\boldsymbol{j};\boldsymbol{N},\boldsymbol{m}) = \frac{\boldsymbol{m}}{\boldsymbol{N}}$ holds, which indicates that revealed sequence includes the specific fragment at the position *a* in source file.

Proof:

$$\sum_{\substack{1\le a\le N\\ 1\le \mathrm{j}\le \mathrm{m}}} \frac{\binom{a-1}{\mathrm{j}-1}\binom{N-a}{\mathrm{m}-\mathrm{j}}}{\binom{N}{\mathrm{m}}} = \sum_{\substack{1\le a\le N\\ 1\le \mathrm{j}\le \mathrm{m}}} \frac{\binom{a-1}{\mathrm{j}-1}\binom{N-1-(a-1)}{\mathrm{m}-1-(\mathrm{j}-1)}}{\frac{N}{m}\binom{N-1}{\mathrm{m}-1}} = \frac{m}{N}\sum_{\substack{1\le a\le N\\ 1\le \mathrm{j}\le \mathrm{m}}} \frac{\binom{a-1}{\mathrm{j}-1}\binom{N-1-(a-1)}{\mathrm{m}-1-(\mathrm{j}-1)}}{\binom{N-1}{\mathrm{m}-1}} = \frac{m}{N} * 1 = \frac{m}{N}$$

2) When *a* is fixed and *j* moves, what is the maximum value of $\mathbf{Succ}(\boldsymbol{a},\boldsymbol{j};\boldsymbol{N},\boldsymbol{m})$?

Assuming *j* is ceiling point that makes $\mathbf{Succ}(\boldsymbol{a},\boldsymbol{j};\boldsymbol{N},\boldsymbol{m})$ to reach its maximum value when *a* is fixed, then following inequation holds

$$\frac{\boldsymbol{Succ}(\boldsymbol{a},\boldsymbol{j}+\mathbf{1};\boldsymbol{N},\boldsymbol{m})}{\boldsymbol{Succ}(\boldsymbol{a},\boldsymbol{j};\boldsymbol{N},\boldsymbol{m})} < \mathbf{1}$$

Then the further calculation of above, we can obtain following as

$$\frac{\boldsymbol{Succ}(\boldsymbol{a},\boldsymbol{j}+\mathbf{1};\boldsymbol{N},\boldsymbol{m})}{\boldsymbol{Succ}(\boldsymbol{a},\boldsymbol{j};\boldsymbol{N},\boldsymbol{m})} = \frac{{}_{a-1}\boldsymbol{C}_j \cdot {}_{N-a}\boldsymbol{C}_{m-j-1}}{{}_{a-1}\boldsymbol{C}_{j-1} \cdot {}_{N-a}\boldsymbol{C}_{m-j}}$$

$$\frac{(\boldsymbol{a}-\boldsymbol{j})(\boldsymbol{m}-\boldsymbol{j})}{(\boldsymbol{N}-\boldsymbol{a}-\boldsymbol{m}+\boldsymbol{j}+\mathbf{1})\boldsymbol{j}} < \mathbf{1} \Rightarrow \mathbf{j} > \frac{\mathbf{am}}{\mathbf{N}} + \mathbf{1}$$

Using the floor function, when it is $\mathbf{j} \ge \left\lfloor \frac{\boldsymbol{am}}{\boldsymbol{N}+\mathbf{1}} \right\rfloor + \mathbf{1}$, then $\boldsymbol{Succ}(\boldsymbol{a},\boldsymbol{j};\boldsymbol{N},\boldsymbol{m}) > \boldsymbol{Succ}(\boldsymbol{a},\boldsymbol{j}+\mathbf{1};\boldsymbol{N},\boldsymbol{m})$ holds. Using ceiling function, when it is $\mathbf{j} \le \left\lceil \frac{\boldsymbol{am}}{\boldsymbol{N}+\mathbf{1}} \right\rceil - \mathbf{1}$, then $\boldsymbol{Succ}(\boldsymbol{a},\boldsymbol{j};\boldsymbol{N},\boldsymbol{m}) < \boldsymbol{Succ}(\boldsymbol{a},\boldsymbol{j}+\mathbf{1};\boldsymbol{N},\boldsymbol{m})$ holds.

As the conclusion, when *a* is fixed and *j* moves ($a\in\{1,N\}$, $j\in\{1,m\}$), the maximum value is given by;

$$\mathbf{Succ}(\boldsymbol{a},\boldsymbol{j};\boldsymbol{N},\boldsymbol{m}) = \left\{ \left\lceil \frac{\boldsymbol{am}}{\boldsymbol{N}+\mathbf{1}} \right\rceil, \left\lfloor \frac{\boldsymbol{am}}{\boldsymbol{N}+\mathbf{1}} \right\rfloor + \mathbf{1} \right\}$$

In other words, $\mathbf{j} = \left\lceil \frac{\boldsymbol{am}}{\boldsymbol{N}+\mathbf{1}} \right\rceil$ makes $\mathbf{Succ}(\boldsymbol{a},\boldsymbol{j};\boldsymbol{N},\boldsymbol{m})$ to reach its maximum value, when *j* moves between 1 to *m*.

3) When *a* moves ($a\in\{1,N\}$), how does $\max_j$=1;:::;*m* Succ(*a; j;N;m*) vary?

The previous calculation of maximum value of $\mathbf{ucc}(\boldsymbol{a},\boldsymbol{j};\boldsymbol{N},\boldsymbol{m})$ ($\mathbf{j} = \left\lceil \frac{\boldsymbol{am}}{\boldsymbol{N}+\mathbf{1}} \right\rceil$) brings a point because of the advantage that attackers can take from it, assuming it is best guess for the attackers. Therefore, following we analysis the various value $\mathbf{Succ}\left(\boldsymbol{a}, \left\lceil \frac{\boldsymbol{am}}{\boldsymbol{N}+\mathbf{1}} \right\rceil; \boldsymbol{N},\boldsymbol{m}\right)$ takes during its extension.

Before the calculation, we will make following analysis;

The curve of binomial distribution shapes symmetric with 1/2 probability and large number of trials as well. However, such large number of trials constitutes significant computational hardship of approximating cumulative distribution of particular discrete random variable (e.g. calculating the probability of having at least 100 heads of certain trials). That is where normal distribution approximates binomial distribution to solve the problem, applying correction for continuity adjustment to discrete random variable for approximating a specific value of continuous random variable – given it is zero in continuous distribution like normal distribution.

Normal equation of random variable value equals:

$$\frac{\mathbf{1}}{\sqrt{\mathbf{2\pi}} * \boldsymbol{\sigma}} \boldsymbol{e}^{\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}$$

where *x*, μ, σ stand for random variable, mean and standard deviation respectively, and (π,*e*) approximately equals (3.14, 2.72).

Thus, to apply normal distribution to approximate binomial distribution, first we use both of the mean and standard deviation of binomial distribution as:

$$\mu = np$$

$$\sigma=\sqrt{np(1-p)}$$

Then by substituting them into the normal equation above, we can write as follows;

$$ {}_nC_k(1-p)^{n-k} \approx \frac{1}{\sqrt{2\pi np(1-p)}} e^{\left(-\frac{(k-np)^2}{2np(1-p)}\right)}$$

By applying the above equation to following respectively

$$ {}_{a-1}C_{j-1}\left(\frac{1}{2}\right)^{a-1}, \quad {}_{N-a}C_{m-j}\left(\frac{1}{2}\right)^{N-a} \quad and \quad {}_{N-1}C_{m-1}\left(\frac{1}{2}\right)^{N-1}$$

, where $j=\left\lceil \frac{am}{N+1} \right\rceil$, we can obtain the corresponding values of each of them respectively as follow;

$$ {}_{a-1}C_{j-1}\left(\frac{1}{2}\right)^{a-1} \approx \frac{1}{\sqrt{2\pi\sqrt{(a-1)/4}}} exp\left(-\frac{(j_a-1-(a-1)/2)^2}{\frac{2(a-1)}{4}}\right) \approx (a-1)^{-1/4}\cdot\frac{1}{\sqrt{\pi}}\left[\exp\left(-2\left(\frac{m}{N}-\frac{1}{2}\right)^2\right)\right]^{a-1}$$

$$ {}_{N-a}C_{m-j}\left(\frac{1}{2}\right)^{N-a} \approx (N-a)^{-1/4} \cdot\frac{1}{\sqrt{\pi}}\left[\exp\left(-2\left(\frac{m}{N}-\frac{1}{2}\right)^2\right)\right]^{N-a} \quad {}_{N-1}C_{m-1}\left(\frac{1}{2}\right)^{N-1} \approx (N-a)^{-1/4}\cdot\frac{1}{\sqrt{\pi}}\left[exp\left(-2\left(\frac{m}{N}-\frac{1}{2}\right)^2\right)\right]^{N-a}$$

$$ {}_{N-1}C_{m-1}\left(\frac{1}{2}\right)^{N-1} \approx (N-a)^{-1/4} \cdot\frac{1}{\sqrt{\pi}}\left[\exp\left(-2\left(\frac{m}{N}-\frac{1}{2}\right)^2\right)\right]^{N-1} \quad {}_{N-1}C_{m-1}\left(\frac{1}{2}\right)^{N-1} \approx (N-1)^{-1/4}\cdot\frac{1}{\sqrt{\pi}}\left[\exp\left(-2\left(\frac{m}{N}-\frac{1}{2}\right)^2\right)\right]^{N-1}$$

As conclusion, we can obtain as following;

$$ \mathrm{Succ}\left(a,\left\lceil\frac{am}{N+1}\right\rceil;N,m\right) = \frac{{}_{a-1}C_{j-1}\cdot {}_{N-a}C_{m-j}}{{}_NC_m} = \frac{m}{N}\cdot\frac{{}_{a-1}C_{j-1}\cdot {}_{N-a}C_{m-j}}{{}_{N-1}C_{m-1}} \approx 1/\sqrt{\pi}\cdot m/N\cdot\left(\frac{N-1}{(a-1)(N-a)}\right)^{1/4}$$

From the equation above, we can assume that the minimum value of $\mathbf{Succ}\left(a,\left\lceil\frac{am}{N+1}\right\rceil;N,m\right)$ can be reached at $a=(N-1)/2$. Fig 2 is simulation result for the value of N=1000 and m=750, in which the result of probability value of $\mathbf{Succ}\left(a,\left\lceil\frac{am}{N+1}\right\rceil;N,m\right)$ is obtained, changing the a for each set of N and m. Thus, we can conclude that the security against identifying the position of any single source file fragment in revealed sequence is uniformly small expect the fragments located in heads and tails.

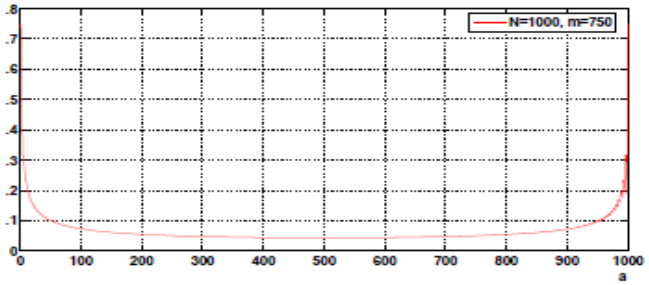

Fig.2. simulation result of N=1000 and m=750.

5. CONCLUSION AND FUTURE WORK

We have investigated some remarkable property of URDA in terms of (k, n) robustness and (k, n) security. We also demonstrated the security property with probability analysis of identifying the fragments of source file is practically small except the fragments positioned at heads and tails. Although this paper does not support conventional (k, n) perfect threshold scheme, it provides acceptable security and time efficiency with highly execution speed.

With current RUDA, users have to archive (n-k+1) times as many files as the original one due to (n-k+1) times redundancy of each fragment of source file. So if a user archives data especially significantly large one, the user would be uncomfortable with the size of the resultant data. And it will also be likely that archiving speed will slow down in real time processing due to the size of the data that user must upload. Therefore as next step we will devote to space efficiency of the algorithm as an expectation of earning user-comfortableness of archiving processes/speed in real time.

[1] [RAID] P.M. Chen, E.K. Lee, G.A. Gibson, Randy H. Katz, and David A. Patterson. Aid: High-performance, reliable secondary storage. ACM Computing Surveys, 26:45.185, 1994.

[2] [Sustor] Yoshino J; Abe H; Kato K;A wide area distributed archival storage for Failure recovery. 2006;NO,86;Page 47-53.

[3] [NAS] "Storage Networking 101", WHITE PAPER

[4] [SAN] "Introduction to Storage Area Networks" redbooks,http://www.redbooks.ibm.com/

[5] Adi Shamir: How to Share a Secret. Communication of ACM, vol.22, no.11, pp.612-613, Nov.1979.7

[6] G.Blakley. safeguarding cryptographic keys.

[7] [Ramp] "Efficient dispersal of information for security, load balancing, and fault tolerance" by Michael Rabin,Journal of the ACM,1989.

[8] Li 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, pages 652.656, 2006.

[9] Li Bai and Xukai Zou. A proactive secret sharing scheme in matrix projection method. International Journal of Security and Networks, 4:In Press, 2009.