



Implicit Security Architecture Framework in Cloud Computing Based on Data Partitioning and Security Key Distribution

S.Hemalatha¹, Dr.R.Manicka Chezian²

¹Ph.D Research Scholar, ²Associate Professor
Research Department of Computer Science
NGM College (Autonomous), Pollachi
Coimbatore, India

Abstract: Cloud computing technology has been visualized as the next-generation architecture of IT venture. In contrast to traditional solutions, where the IT services are under proper physical, logical and personnel controls, Cloud Computing shifts the application software and databases to the large data centers, where the management of the data and services may not be fully reliable. This unique attribute, however, impose many new security challenges which have not been well understood. In this paper, we focus on security based on cloud data storage, which has always been an important aspect of quality of service. To ensure the correctness of user's data in the cloud, we suggest an effective and flexible distributed scheme with two salient features, contrasting to its predecessors. We illustrate the use of a data partitioning method for implementing such security involving the roots of a polynomial in restricted field. The partitions are stored on randomly chosen servers on the network and they need to be retrieved to reconstruct the original data. Data reconstruction requires access to each server, the knowledge of the servers on which the partitions are stored and login password. Several variations of this scheme are described, which include the implicit storage of encryption keys rather than the data, and where a subset of the partitions may be brought together to reconstruct the data.

Keywords: Cloud Computing; Data Partitioning; Decryption; Encryption; Explicit Security; Implicit Security; Key Distribution

I. Introduction

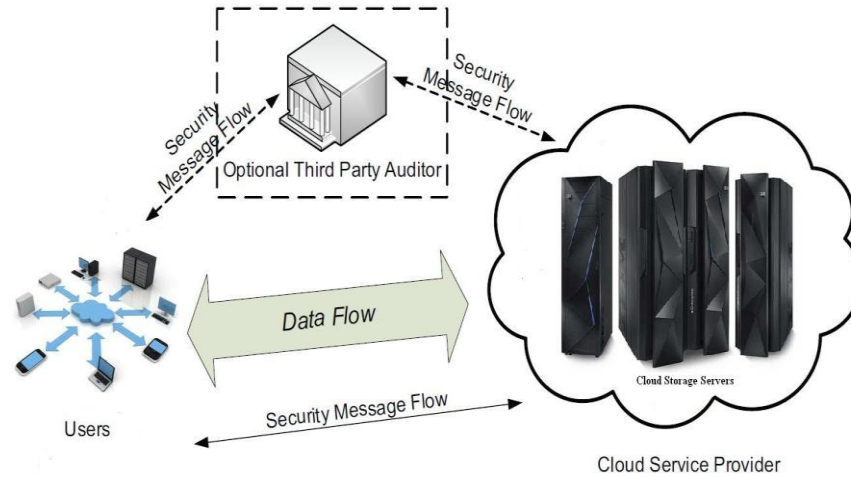
Numerous trends are opening up the era of Cloud Computing, which is an Internet based development and use of computer technology. The forever cheaper and more powerful processors, together with the software as a service (SaaS) computing architecture, are transforming data centers into pools of computing service on a massive scale. The increasing network and reliable flexible network connections make it even possible that users can subscribe high quality services from data and software that reside exclusively on remote data centers.

Moving data into the cloud provides great convenience to users since they don't have to care about the complexities of direct hardware management. The pioneer of Cloud Computing vendors, Amazon Simple Storage Service (S3) and Amazon Elastic Compute Cloud (EC2) [1] are both well known examples. At the same time as these internet based online services do provide huge amounts of storage space and customizable computing resources, this computing platform shift, however, is eliminating the responsibility of local machines for data maintenance at the same time. As a result, users are at the compassion of their cloud service providers for the availability and integrity of their data. Latest downtime of Amazon's S3 is such an example [2].

From the viewpoint of data security, this has always been an important aspect of quality of service; Cloud Computing inevitably poses new challenging security threats for number of reasons. Initially, traditional cryptographic primitives for the purpose of data security protection cannot be directly implemented due to the users' loss control of data under Cloud Computing. Therefore, verification of correct data storage in the cloud must be carried out without explicit knowledge of the whole data. Taking into consideration various kinds of data for each user stored in the cloud and the demand of long term continuous assurance of their data safety, the problem of validating correctness of data storage in the cloud becomes even more challenging. Next, Cloud Computing is not just a third party data warehouse. The data stored in the cloud may be regularly updated by the users, including insertion, deletion, modification, appending, reordering, etc. To ensure storage correctness under dynamic data update is hence of principal importance. However, this dynamic feature also makes traditional integrity insurance techniques useless and requires new solutions. Last but not the least, the deployment of Cloud Computing is powered by data centers running in a synchronized, cooperated and distributed manner. Individual user's data is stored redundantly in multiple physical locations to further reduce the data integrity threats. Therefore, distributed protocols for storage correctness assurance will be of most importance in achieving a secure and robust cloud data storage system in the real world. However, such important area remains to be fully investigated in the literature.

In recent times, the importance of ensuring the remote data integrity has been highlighted by the following research works [3]–[7]. These techniques, while can be useful to guarantee the storage correctness without having users possessing data, cannot address all the security threats in cloud data storage, since they are all focusing on single server state and most of them do not consider dynamic data operations. As a complementary approach, researchers have also proposed distributed protocols [8]–[10] for ensuring storage appropriateness across multiple servers or peers. Yet again, none of these distributed schemes is aware of dynamic data operations. As a result, their applicability in cloud data storage can be drastically restricted.

Figure 1 Cloud data storage architecture



In this paper, we propose an effective and flexible distributed scheme with implicit dynamic data support to ensure the correctness of users' data in the cloud. We have illustrated an implicit security architecture suited for the application of online storage. In this method data is partitioned in such a way that each partition is implicitly secure and does not need to be encrypted. These partitions are stored on different servers on the network which are known only to the user. Reconstruction of the data requires access to each server and the knowledge as to which servers the data partitions are stored. Several variations of this scheme are described, which include the implicit storage of encryption keys rather than the data, and where a subset of the partitions may be brought together to reconstruct the data. Our contribution can be summarized as the following aspects:

- Compared to many of its predecessors (traditional or explicit) approach to securing data is to store and back it up on a single server and allow access upon the use of passwords that are needed to be frequently changed, this scheme is different from them.
- Extensive security and performance analysis shows that the projected scheme is highly efficient and resilient against malicious data modification attack, Byzantine failure, and even server colluding attacks.
- One may incorporate further security within the system by using puzzles [11] or otherwise by another layer that increases the space that the intruder must search in order to break the system.

The rest of the paper is structured as follows. Section II introduces the system model, adversary model and our design goals. Then we provide the overview of the proposed security framework and detailed description of our scheme in Sections III and IV respectively. Section V gives the details of addressing the data partitions being made followed by Section VI which overviews the second part of this paper, namely, security by key distribution. Finally, Section VII gives the concluding remark of the whole paper.

II. Problem Statement

2.1 System structure

A representative network architecture for cloud data storage is illustrated in Figure 1 shown above. Three different network entities can be recognized as follows:

- User: users, who have data to be stored in the cloud and depend on the cloud for data computation, consist of both individual consumers and organizations.
- Cloud Service Provider (CSP): a CSP, who has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates live Cloud Computing systems.
- Third Party Auditor (TPA): an optional TPA, who has expertise and capabilities that users may not have, is trusted to evaluate and interpret risk of cloud storage services on behalf of the users upon request.

In cloud data storage, a user stores his data through a CSP into a set of cloud servers, which are running in a synchronized, cooperated and distributed manner. Data redundancy can be engaged with technique of erasure-correcting code to further tolerate faults or server crash as user's data grows in size and importance. Subsequently, for application purposes, the user interacts with the cloud servers via CSP to access or retrieve his data. In few cases, the user may need to perform block level operations on his data. The most general forms of these operations we consider are block update, delete, insert and append.

As users no longer possess their data locally, it is of significant importance to guarantee users that their data are being correctly stored and maintained. That is, users should be outfitted with security means so that they can make continuous correctness assurance of their stored data even without the existence of local copies. In case those users do not necessarily have the time, feasibility or resources to supervise their data, they can hand over the tasks to an optional trusted TPA of their respective choices. In our proposed model, we assume that the point-to-point communication channels between each cloud server and the user is reliable and authenticated, which can be achieved in practice with little overhead. Note down that we don't address the issue of data privacy in our paper, as in Cloud Computing, data privacy is orthogonal to the problem we learn here.

2.2 Adversary model

Security threats expressed by cloud data storage can come from two different kinds of sources. On one hand, a CSP can be untrusted, self-interested and possibly malicious. Not only does it craving to move data that has not been or is rarely accessed to a lower tier of storage than agreed for monetary reasons, but it may also try to hide a data loss incident due to management errors, Byzantine failures and so on. On the other hand, there may also exist an economically provoked adversary, who has the capability to compromise a number of cloud data storage servers in different time intervals and subsequently is able to modify or delete user's data while remaining undetected by CSPs for a certain period. Specifically, we consider two types of adversaries with different levels of capability in this paper as described below:

Weak Adversary: The opponent is interested in corrupting the user's data files stored on individual servers. Once a server is encompassed, an adversary can pollute the original data files by modifying or introducing its own fraudulent data to prevent the original data from being retrieved by the user.

Strong Adversary: This is the most horrible scenario, in which we assume that the adversary can compromise all the storage servers so that he can intentionally modify the data files as long as they are internally reliable. In fact, this is equivalent to the case scenario where all servers are colluding together to hide a data loss or corruption incident.

2.3 Design goals

To guarantee the security and dependability for cloud data storage under the above mentioned adversary model, we aim to design efficient mechanisms for dynamic data verification and operation and achieve the following goals: (1) Data partitioning: stored data is partitioned into two or more pieces and stored at randomly chosen places on the network that are known only to the owner of the data. (2) Distribution: distributing its parts over various servers and these partitions is based on a generalization of the use of 3 or 9 roots of a number in a cubic transformation [12]. (3) Secret sharing scheme: based on Shamir's secret sharing scheme [13] which takes the advantage of polynomial interpolation and maps the data as roots of a polynomial on the x-axis. (4) Reconstruction of data: requires access to each server and the knowledge as to which servers the data partitions are stored.

III. Overview of Problem Statement

Securing data stored on distributed servers is of primary importance in cloud computing. The conventional (explicit) approach to securing data is to store and back it up on a single server and allow access upon the use of passwords that are needed to be frequently changed. But there is a tendency among users to keep passwords simple and memorable [14,15,16] leading to the likelihood of brute force attacks. Moreover since the data on the Web is archived, keys that provide adequate encryption today are probable to be broken in the future. As a result, explicit security architecture may not be sufficient for many applications.

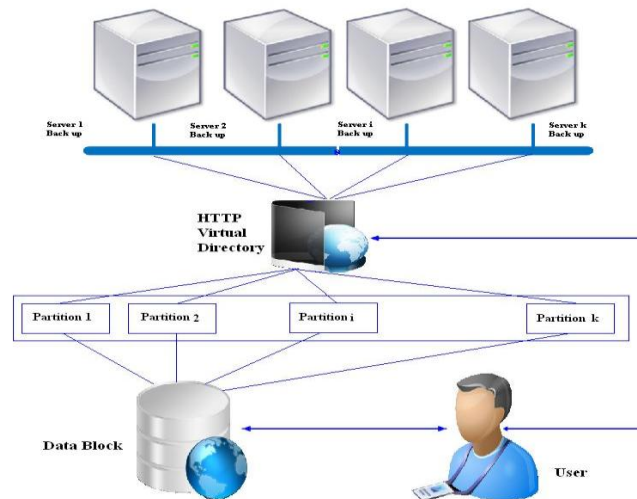
To go ahead of the present approach, one may integrate further security within the system by using puzzles [11] or otherwise by another layer that increases the space that the intruder must search in order to break the system. Here, we suggest an implicit security architecture in which security is distributed between many entities. In comparison with hash-based distributed security models for wireless sensor networks [17–21], we consider a more general method of data partitioning. In this approach, stored data is partitioned into two or more pieces and stored at randomly chosen places on the network that are known only to the owner of the data. Access to these pieces not only depends on the knowledge of a password but also on the knowledge of where the pieces are stored. The division of data is performed in such a way that the knowledge of all the pieces is required to reconstruct the data and that none of the individual pieces reveals any useful information. In circumstances

where one or more pieces may be at the danger of being lost or inaccessible due to system or network failure, one may employ schemes that can reconstruct the data from a subset of original pieces. Formally, our projected scheme consists of two parts – the first part is a (k,k) partitioning scheme where all the k partitions are required to reconstruct the data and the second part extends the first part of the scheme to a (k,p) partitioning scheme, where $k \leq n$ and $k \geq 2$, i.e. only k out of n partitions are required to reconstruct the data.

A number of methods have been proposed in the communications perspective for splitting and haring of decryption keys [22–24]. These schemes descend under the category of “secret sharing schemes”, where the decryption key is considered to be a secret. Motivated by the need to have an analog of the case where several officers must concurrently use their keys before a bank vault or a safe deposit box can be opened, these schemes do not consider the requirement of data protection for a single party. Further, in any secret sharing scheme it is understood that the encrypted data is stored in a secure place and that none of it can be compromised without the decryption key. Some schemes that directly apply secret sharing for distributed data storage in sensor networks have been proposed [25–27], but have a complexity of $O(n \log^2 n)$ at best, whereas our scheme can achieve linear complexity. Other schemes aim at sharing multiple secrets [28] but have a complexity of $O(n^2)$ or produce group keys for group sharing of information [29]. Certain probability models have been projected for reconstructing secret sharing over the Internet [30], but only focus on strategies for share distribution over the network and do not provide methods for data partitioning.

In this paper, we therefore protect data by distributing its parts over various servers. The idea of doing these partitions is a generalization of the use of 3 or 9 roots of a number in a cubic transformation [12]. The scheme we propose is simple and easily implementable. We would like to stress that the presented scheme is different from Shamir’s secret sharing scheme [13] which takes the advantage of polynomial interpolation and maps the secret on the y-axis; whereas we map the data as roots of a polynomial on the x-axis.

Figure 2 Implicit and explicit security architectures



IV. Proposed (k,k) Data Partitioning Scheme

We consider the model of Figure 2 to illustrate the difference between explicit and implicit security. We need to partition data and send it to randomly chosen servers. We now describe our data partitioning scheme. By the elementary theorem of algebra, every equation of degree k has k roots. We use this fact to partition data into k partitions such that each of the partition is stored on a different server. No explicit encryption of data is necessary to secure each partition. The partitions in themselves do not disclose any information and hence are implicitly secure. Only when all the partitions are brought together is the data discovered.

Consider an equation of order k

$$x^k + a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \dots + a_1x + a_0 = 0 \quad (1)$$

Eq. (1) has k roots denoted by $\{r_1, r_2, \dots, r_k\} \subseteq \{\text{set of complex numbers}\}$ and can be rewritten as

$$(x - r_1)(x - r_2) \dots (x - r_k) = 0 \quad (2)$$

In cryptography, it is more convenient to use the finite field $\mathbb{Z}_p$ where p is a large prime. If we replace a_0 in (1) with the data $d \in \mathbb{Z}_p$ that we wish to partition then,

$$x^k + \sum_{i=1}^{k-1} a_{k-i} x^{k-i} + d \equiv 0 \pmod{p} \quad (3)$$

where $0 \leq a_i \leq p-1$ and $0 \leq d \leq p-1$. (Note that one may alternatively use $-d$ in (3) instead of d .) This may be rewritten as

$$\prod_{i=1}^k (x - r_i) \equiv 0 \pmod{p} \quad (4)$$

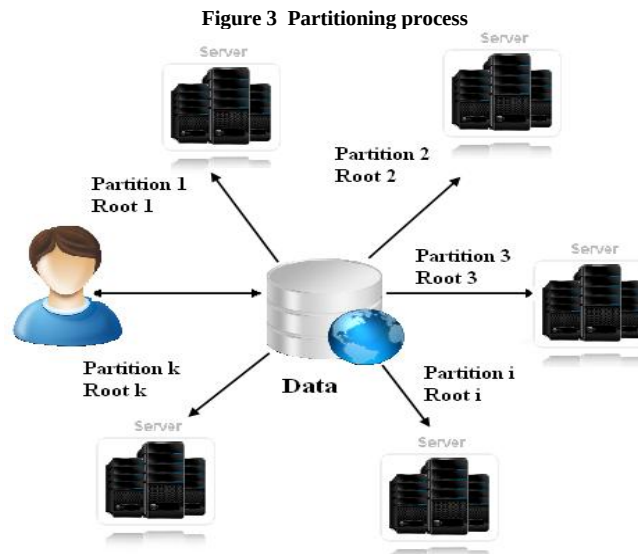
where $1 \leq r_i \leq p-1$. The r_i are the partitions (see Figure 3). For example, if the data needs to be divided into two parts then an equation of second degree is chosen and the roots computed. If we represent this general equation by

$$x^2 + a_1 x + d \equiv 0 \pmod{p} \quad (5)$$

then the two roots can be calculated by solving the following equation modulo p :

$$x = \frac{-a_1 \pm \sqrt{a_1^2 - 4d}}{2} \quad (6)$$

which has an solution in $\mathbb{Z}_p$ only if the square root exists modulo p . If the square root does not exist then a different value of a_1 needs to be chosen.



V. Dealing with the data partitions

The preceding sections consider the security of the proposed scheme when prime p and composite n are public knowledge. However, there is nothing that compels the user to disclose the values of p and n are secret values then the partitions may be stored in the structure of an “encrypted link list”. We describe encrypted link list as a link list in which every pointer is in encrypted form and in order to find out which node the present node points to, the user needs to decrypt the pointer which can be done only if definite secret information is known.

If we assume that p and n are public values then the pointer can be so encrypted that each decryption either directs to multiple addresses or depends on the knowledge of the factors or both. Only the authentic user will know which of the multiple addresses is to be selected.

On the other hand, one may use a random number generator and generate a random sequence of server IDs using a secret seed. One way to produce this seed may be to find the hash of the original data and use it to seed the generated sequence and keep the hash secret.

VI. Security by Key Partitioning and Distribution

In few cases, when there is a large amount of data, a user may find it ineffective to create data partitions and distribute them over the network but may wish to encrypt the data and store it on a single server which he trusts and keep encryption key secret. Roughly always, encryption keys are very large random numbers and cannot be memorized. Therefore, the user may create partitions of the key and spread them over the network. This approach may be more efficient, if not more secure, than creating data partitions of enormous amounts of data. Moreover, the access to the servers on which the key partitions are stored may be password restricted.

We stress that this scenario is different from the secret sharing scenario where multiple parties are involved given that our case involves data security for a single party. Keys may be partitioned using the scheme presented in the preceding sections where the data is replaced by the key.

VII. Conclusion

In this paper, we investigated the problem of data security in cloud data storage, which is fundamentally a distributed storage system. To guarantee the correctness of user's data in cloud data storage, we proposed an effective and flexible distributed scheme. We have described an implicit security architecture suited for the application of online storage and in this scheme data is partitioned in such a way that each partition is implicitly secure and does not need to be encrypted. These partitions are stored on different servers on the network which are known only to the user. Reconstruction of the data requires access to each server and the knowledge as to which servers the data partitions are stored. Several variations of this scheme are described, which include the implicit storage of encryption keys rather than the data, and where a subset of the partitions may be brought together to reconstruct the data. The proposed scheme is also of potential use in sensor networks. To protect sensitive information on sensors that are physically compromised, one may partition the data and store these parts on other sensors in the network. The key that is used to find the location of these parts is encrypted and this encryption is changed from time to time. This provides solutions that are more general than other distributed security models used in sensor networks [17]. We believe that data storage security in Cloud Computing, an area full of challenges and of principal importance, is still in its childhood now, and many research problems are yet to be identified. We envisage several possible directions for future research on this area.

VIII. References

- [1] Amazon.com, 2008. Amazon Web Services (AWS). Online at <http://aws.amazon.com>
- [2] N. Gohring, 2008. Amazon's S3_down_for_several_hours. Online at http://www.pcworld.com/businesscenter/article/142549/amazons_s3down_for_several_hours.html.
- [3] A. Juels and J. Burton S. Kaliski, 2007. PORs: Proofs of Retrievability for Large Files. Proceedings. of CCS '07, 584–597.
- [4] H. Shacham and B. Waters, 2008. Compact Proofs of Retrievability. Proceedings of Asiacrypt '08. Dec. 2008.
- [5] K. D. Bowers, A. Juels, and A. Oprea, 2008. Proofs of Retrievability: Theory and Implementation. Cryptology ePrint Archive, Report 2008/175. <http://eprint.iacr.org/>.
- [6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, 2007. Provable Data Possession at Untrusted Stores. Proceedings of CCS '07, 598–609.
- [7] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, 2008. Scalable and Efficient Provable Data Possession. Proceedings of SecureComm '08, 1–10.
- [8] T. S. J. Schwarz and E. L. Miller, 2006. Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage. Proceedings of ICDCS '06, 12–12.
- [9] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, 2003. A Cooperative Internet Backup Scheme. Proceedings of the 2003 USENIX Annual Technical Conference (General Track), 29–41.
- [10] K. D. Bowers, A. Juels, and A. Oprea, 2008. HAIL: A High-Availability and Integrity Layer for Cloud Storage. Cryptology ePrint Archive, Report 2008/489. <http://eprint.iacr.org/>.
- [11] S. Kak, 1984. On the method of puzzles for key distribution, International Journal of Computer and Information Science 14, 103–109.
- [12] S. Kak, 2007. A cubic public-key transformation, Circuits, Systems and Signal Processing 26, 353–359.
- [13] A. Shamir, 1979. How to share a secret, Communication of ACM 22 (11), 612–613.
- [14] E. Gheringer, 2002. Choosing passwords: security and human factors. Proceedings of IEEE 90, 369–373.
- [15] R. Proctor, M. Lien, K. Vu, G. Salvendy, 2002. Improving computer security for authentication of users: influence of proactive password restrictions, Behavior Research Methods, Instruments and Computers 34, 163–169.
- [16] S. Riley, 2006. Password security: what users know and what they actually do, Usability News 8.1.
- [17] N. Subramanian, C. Yang, W. Zhang, 2007. Securing distributed data storage and retrieval in sensor networks, Pervasive and Mobile Computing 3 (6), 659–676.
- [18] G. Wang, W. Zhang, G. Cao, T.L. Porta, 2003. On supporting distributed collaboration in sensor networks. Proceedings of IEEE Military Communications Conference.
- [19] F. Ye, H. Luo, J. Cheng, S. Lu, L. Zhang, 2002. A two-tier data dissemination model for large-scale wireless sensor networks. Proceedings of ACM International Conference on Mobile Computing and Networking, 148–159.
- [20] W. Zhang, G. Cao, T.L. Porta, 2003. Data dissemination with ring-based index for sensor networks. Proceedings of IEEE International Conference on Network Protocol.
- [21] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, S. Shenker, 2003. Difs: a distributed index for features in sensor networks, Ad Hoc Networks 1, 333–349.
- [22] A. Shamir, 1979. How to share a secret. Communication of ACM 22 (11), 612–613.
- [23] D. Denning, 1982. Cryptography and Data Security. Addison-Wesley Publishing Company.
- [24] A. Renvall, C. Ding, 1996. A nonlinear secret sharing scheme, Information Security and Privacy, LNCS 1172, 56–66.
- [25] T. Claveirole, M. de Amorim, M. Abdalla, Y. Viniotis, 2008. Securing wireless sensor networks against aggregator compromises. Communications Magazine IEEE 46 (4), 134–141.
- [26] A. Eldin, A. Ghalwash, H. ElShandidy, 2008. Enhancing packet forwarding in mobile ad hoc networks by exploiting the information dispersal algorithm, Communications, Computers and Applications. MIC-CCA 2008. Mosharaka International Conference, 20–25.
- [27] T. Yuan, S. Zhang, 2008. Secure fault tolerance in wireless sensor networks in CITWORKSHOPS'08. Proceedings of the 2008 IEEE Eighth International Conference on Computer and Information Technology Workshops, 477–482.
- [28] M.H. Dehkordi, S. Mashhadi, 2008. New efficient and practical verifiable multi-secret sharing schemes, Information Sciences 178 (9), 2262–2274.
- [29] C. Kuang-Hui, C. Wen-Tsuen, 1992. A new group key generating model for group sharing, Information Sciences 64 (1-2), 83–94.
- [30] C.-Y. Lee, Y.-S. Yeh, D.-J. Chen, K.-L. Ku, 1999. A probability model for reconstructing secret sharing under the internet environment, Information Sciences 116 (2-4), 109–127.