

# Multilevel Threshold Secret Sharing in Distributed Cloud

Doyel Pal<sup>1</sup>(✉), Praveenkumar Khethavath<sup>2</sup>, Johnson P. Thomas<sup>1</sup>,  
and Tingting Chen<sup>3</sup>

<sup>1</sup> Computer Science Department, Oklahoma State University,  
Stillwater, OK 74075, USA  
{doyelp,jpt}@cs.okstate.edu

<sup>2</sup> Mathematics, Engineering and Computer Science Department, LaGuardia  
Community College, CUNY, Longisland City, NY 11101, USA  
pkhethavath@lagcc.cuny.edu

<sup>3</sup> Computer Science Department, California State Polytechnic University,  
Pomona, CA 91768, USA  
tingtingchen@csupomona.edu

**Abstract.** Security is a highlighted concern in cloud and distributed cloud systems. Threshold secret sharing scheme is a widely used mechanism to secure different computing environments. We split secret into multiple shares and store them in different locations using threshold secret sharing scheme. In this paper we propose a multilevel threshold secret sharing scheme to enhance security of secret key in a distributed cloud environment. We create replicas of secret shares and distribute them among multiple resource providers to ensure availability. We also introduce dummy shares at each resource provider to realize the presence of any outside attacker. Our experiment results show that our scheme is feasible and secure.

## 1 Introduction

The Distributed cloud [16] makes use of resources provided by users in a P2P manner. In the distributed cloud resources are virtualized.. The existing cloud uses huge data centers where resources are provided using virtualization techniques. Some of the voluntary computing systems such as BOINC [14], SETI [15] and Planetlab [20] uses resources provided by users, but these are managed by a central entity and are completely different from either a centralized cloud or the distributed cloud. Unlike users of the existing cloud, who don't have control over the resources, in the distributed cloud users can choose resources of their choice. Moreover, in the distributed cloud, resources are provided by users. Since users of the distributed cloud have more control over resource selection, they can perform strong encryption to secure their data. The only issue will be managing these keys. We use secret sharing mechanism in this paper to protect secret keys in the distributed cloud.

Secret sharing schemes can mitigate the risks of managing these keys. Secret sharing schemes are used in many cryptographic protocols. Threshold secret

sharing scheme is the most widely used technique. The main ideas behind the threshold secret sharing schemes was introduced by Shamir [1] and Blakley [2]. In the threshold secret sharing mechanism, information is split into multiple shares and these shares are distributed among multiple users. The only way to retrieve the information back is to have all the shares or a qualified number of shares available. This qualified number is the threshold. This means that any number of shares less than the threshold cannot retrieve the information. Shamir's secret sharing scheme is based on polynomial interpolation whereas Blakley's secret sharing scheme is based on geometry. Several secret sharing schemes have been introduced, e.g., threshold secret sharing scheme, ideal secret sharing, linear secret sharing, multi linear secret sharing, [1, 2, 4, 5] etc. The Asmuth and Bloom secret sharing scheme [17] uses the Chinese remainder theorem to reconstruct shares. Multi linear secret sharing schemes [18] use monotone span programs. In these schemes, secret is a sequence of elements from a finite field and the share is derived using the secret and some random elements from the field. If the secret is of size one i.e. has only one element it is called linear.

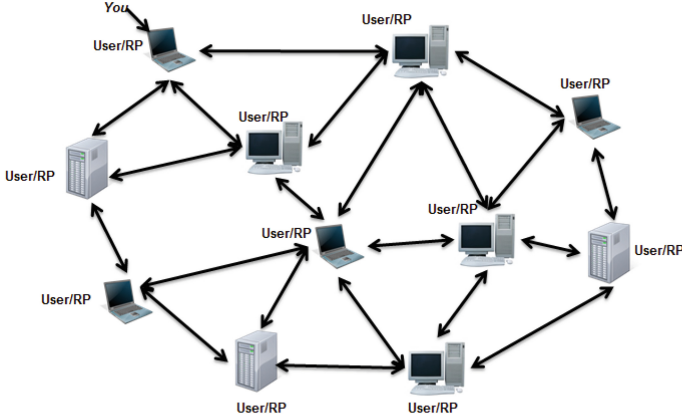
In this paper we propose a multilevel threshold secret sharing scheme to provide security of secret key in the distributed cloud. In the first level of the proposed secret sharing scheme we split the secret key into multiple shares and distribute them among multiple resource providers. The second level consists of splitting each share into multiple numbers of sub-secrets at each resource provider. To make our scheme secure we make the determination of threshold value at the second level dynamic. We generate a share pool which is used to determine the number of shares at each resource provider using Chinese Remainder Theorem (CRT) [13]. The advantage of our proposed scheme is that in the distributed cloud, users provide resources and therefore users do not need to invest in the commercial cloud (e.g., Amazon AWS [22], EC2 [23], Windows Azure [24] etc.). Security in the distributed cloud has not been studied much and our scheme provides a way to get a secure way to protect the secret key. We perform the multilevel threshold secret sharing mechanism on the secret key itself as key plays an important role to encrypt or decrypt the file content. In our proposed scheme the secret is distributed over multiple resource providers therefore an attacker cannot have the original secret until and unless all the participating resource providers are compromised. We generate the replica of each secret share to ensure the fact that if a provider is compromised then that particular share is available from other providers. Moreover the user can revoke the access of any resource provider if the provider is compromised. We increase the chance to detect if a provider is compromised by introducing dummy key shares at each resource provider.

The rest of the paper is organized as follows. In Sects. 2 and 3, we describe related work and present the problem statement respectively. In Sect. 4, we explain our proposed solution. In Sect. 5, we discuss the performance evaluations of our approach and in Sect. 6 we provide our conclusions.

## 2 Related Work

In this paper we use Shamir's threshold secret sharing scheme [1]. [1] is an ideal perfect secret sharing scheme as the size of shares are exactly same with the size of each share. A lot of work [5–7] has been developed based on [1]. A threshold ideal secret sharing has been proposed in [6] which uses only XOR operations to make shares and to recover the secret. The authors also show that this scheme is perfect and faster with reduced storage usage and computational cost compared to [1]. In [7] a multilevel threshold secret sharing scheme has been proposed with two modifications of [1]. The first modification allow the shareholders to keep both x coordinate and y- coordinate along with the share of secret and in the second modification a polynomial degree larger than the threshold value is used by the dealers to generate the share. In our scheme we perform splitting the secret twice, once by the user and the other by the resource provider. At resource provider along with the shares we also store dummy shares which improves security. Another type of secret sharing scheme, Linear secret sharing scheme [1, 2, 5] is based on linear algebra and it contains super polynomial lower bounds on the share size. Though [1] can only realize the access structure of the secret sharing scheme, [5] can design a secret sharing scheme from any given access structure. A multi linear secret sharing scheme which uses super polynomial bounds on the share size has been proposed in [4]. In [3] the authors introduce a hierarchical threshold secret sharing scheme. In this scheme secret is shared in a hierarchical structure among groups of participants which is further partitioned into levels.

Security is an important concern for cloud architecture as cloud storage is vulnerable to security threats. Protecting data storage and key management are two of the most important concerns among them. [8–12] concentrate on the cloud storage security and the secure key management scheme in different cloud environments. Cloudstash [8] concentrates on providing the security of the cloud storage. It considers the file as secret and applies secret sharing scheme directly on the multiple shares of secrets. This scheme hashes and signs each share of file and then distributes these signed multi shares into multi clouds. To overcome the limitations of single domain cloud a new second layer in the dependable cloud computing stack named Intercloud has been introduced in [9]. This scheme performs symmetric encryption on the data and splits the key into shares using secret sharing scheme. It attaches the key shares as metadata to the pieces of data and distributes them to the clouds. The N cloud scheme [10] propose an improved cloud storage scheme in terms of availability, performance and confidentiality in cloud. It splits the file in many chunks and uploads the encrypted file chunks in geographically separated cloud storage's in parallel. It also replicates the chunks in non overlapping manner into many cloud storages. To reconstruct the file it downloads the chunks, decrypts them and reassembles them back to a file. This technique manages the encryption keys at client side and does not save them in clouds. To overcome the limitation of single cloud, the DepSky system [11] builds a cloud of clouds named Intercloud, on top of a set of storage clouds by using a combination of diverse commercial clouds. It uses the combination of



**Fig. 1.** Distributed cloud model

Byzantine quorum system protocols, cryptographic secret sharing, erasure codes and diversity of several clouds to improve the availability and confidentiality provided by commercial storage cloud. The CloudSeal scheme [12] provides an end to end content confidentiality protection mechanism for large scale content storage. It uses integrated symmetric encryption, threshold secret sharing, and proxy based re- encryption scheme to protect the content and to manage the user access. It also supports forward and backward security. In our multi-level threshold secret sharing scheme we provide security to protect secret key in the distributed cloud.

### 3 Problem Statement

In the distributed cloud model [16,21], users do computation and store data in resources provided by other users as shown in Fig. 1. Since users use resources provided by other users, there are obvious security concerns. Dependable Storage in the Intercloud [11], provides reasons why single cloud is not secure, namely, we need to have more than one cloud to make the system more secure. The Distributed cloud model by itself solves the problem of single domain cloud as it has more than one user who will be acting as a cloud provider. In distributed cloud, protecting data stored on other users' resources poses security issues, which is handled using standard encryption techniques, which in turn leads to key management issues and insider attacks. A resource provider can get access to the data stored on his resources with ease and can use brute force attacks on it. Security of data storage currently depends on how strong encryption keys a user has used or how effective the key management schemes used are. So instead of having a single key that gives access to the encrypted files, we propose using multiple shares. In our proposed secret sharing mechanism in the distributed cloud we use secure mechanisms to enhance the security of secret key shares.

We have a share pool that determines the number of shares for a particular key share.

In the distributed cloud, since users can join and leave the cloud, we need to make sure this resource pool is always available. So we make duplicates of this pool and store them for multiple resource providers in all clusters in the distributed cloud.

## 4 Proposed Solution

### 4.1 Preliminaries

In this section we discuss the basic preliminaries we use in this paper, that is, Shamir's threshold secret sharing scheme and Chinese Remainder Theorem.

**Shamir's Threshold Secret Sharing.** In this paper we use Shamir's  $(k, n)$  threshold secret sharing scheme [1]. Shamir's  $(k, n)$  threshold secret sharing scheme is based on polynomial interpolation. A secret  $S$  can be shared by  $n$  number of users and can be reconstructed as well, if the number of shares to reconstruct exceeds some threshold value  $k$ . This scheme uses a polynomial function of order  $(k - 1)$  which can be constructed as,

$f(x) = d_0 + d_1x + d_2x^2 + \dots + d_{k-1}x^{k-1}$  Here  $d_0$  is the secret  $S$ . Given any  $k$  shares  $\langle x_0, f(x_0) \rangle, \dots, \langle x_{(k-1)}, f(x_{(k-1)}) \rangle$  the secret  $S$  can be reconstructed using Lagrange Interpolation formula as follows:

$$\sum_{i=0}^{k-1} \left( \prod_{j \neq i} \frac{x_j}{x_j - x_i} \right) f(x_i)$$

**Chinese Remainder Theorem.** We use the Chinese Remainder Theorem (CRT) [13] to generate the threshold value at each resource provider at the second level of the threshold secret sharing scheme. Given the following system of equations,

$$\begin{aligned} x &= a_1 \bmod p_1 \\ x &= a_2 \bmod p_2 \\ &\dots\dots\dots \\ x &= a_k \bmod p_k \end{aligned}$$

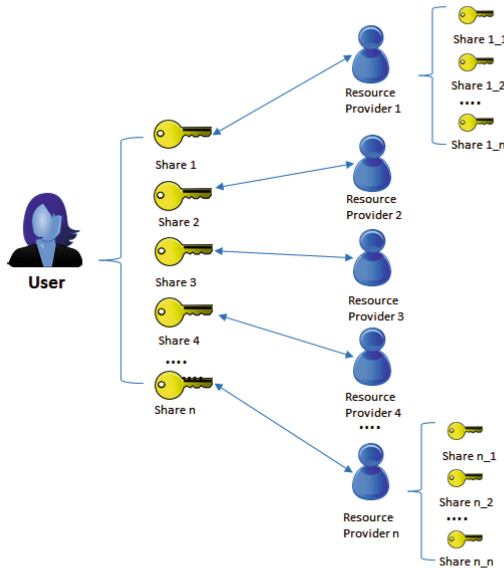
There is one unique solution as

$$x = \sum_{(i=1)}^k \frac{P}{p_i} y_i a_i \bmod P$$

where  $P = p_1 p_2 \dots p_k$  and  $\frac{P}{p_i} y_i \bmod p_i = 1$ , if all moduli are pairwise coprime, i.e.,  $\gcd(p_i, p_j) = 1$  for every  $i \neq j$ .

## 4.2 Proposed Scheme

The Distributed cloud is formed using the resource provided by users and can provide resource for free to everyone who is part of the system. The Distributed cloud makes use of virtualization to allocate resources to multiple users and shares available resources efficiently and it also avoids single point of failure. In this paper we propose a scheme to protect the secret key using a multilevel threshold secret sharing mechanism shown in Fig. 2 in the distributed cloud environment. Our proposed scheme encrypts the file using the secret key before distributing the key shares among participant resource providers whom we assume to be honest. At the first level the user splits the key and distributes the shares among resource providers. Instead of attaching key shares as metadata to the pieces of data [9] we split the key shares at each resource provider again into multiple shares in the second level. The second level of our mechanism improves the security since to get the original secret the attacker has to have all the shares from the two levels. We generate the threshold value in the second level dynamically which enhances the security as the attacker cannot know about the threshold value beforehand. In addition to that at each resource provider we create dummy keys which increases the probability of knowing if a resource provider is compromised by any attacker. In Algorithm 1 we ensure the security of the secret key in a distributed cloud environment. To do that we use the multilevel threshold secret sharing scheme  $(t, n)$  in which the first level uses the threshold  $t < n$  and the second level uses the threshold  $t = m$ .



**Fig. 2.** Proposed multilevel threshold secret sharing scheme

First the user splits the secret key into  $n$  number of shares  $S_i$ , where  $i \in (1, n)$  and distribute them among all resource providers (Algorithm 1.2. line 1 to 7). In this level the threshold value  $t < n$  which indicates that to reconstruct the secret key at least  $t$  number of shares would be required. Each share of secret  $S_i$  is replicated into  $k$  numbers so that if one resource provider goes offline or compromised then that share can be accessed from other resource provider. At each resource provider we further split the share  $S_i$  into  $m$  number of shares  $S_{ij}$  (Algorithm 1.2. line 12). At this level, i.e., second level the threshold value  $m$  is generated dynamically. To determine the number of shares  $m$  each resource provider  $R_i$ ,  $i \in (1, n)$  selects a  $(P_i, N_i)$  pair from the share pool SP (Algorithm 1.2 line 9 to 12). The share pool is created beforehand (Algorithm 1.1. Generate share pool SP). The user saves the pair  $(i, P_i)$  for each provider  $R_i$  and the provider saves  $N_i$ . We intend to have multiple share pools and place one or two of them in each cluster of the distributed cloud. The CRT solution generates a number  $m$  which decides the number of shares to split and reconstruct in the second level. At each resource provider at least  $j$  number of dummy shares  $S_{ij}^D$ , where  $i \in (1, n)$  and  $j \geq m$ , are generated (Algorithm 1.2 line 13). Whenever a resource provider is compromised, the user revokes the access of that particular resource provider. We intend to have a greater number of dummy shares than secret shares, i.e.,  $j \geq m$ , so that if any outside attacker tries to get the share, the probability that he ends up with the dummy share instead of a real share is greater than or equal to 0.5. This helps the user to take action (e.g., revoke the access of that resource provider) accordingly when some attacker selects a dummy key. To reconstruct the key sub shares, each resource provider need to have the  $P_i$  from the user to generate the threshold value  $m$  (Algorithm 1.3. line 2). The user reconstructs the secret  $S$  from  $t$  numbers of  $S_i$  shares.

## Algorithm 1

**Input:** The secret key  $S$ ,  $n$  number of resource providers in distributed cloud, threshold value  $t \leq n$ .

### *Algorithm 1.1. Generate share pool SP*

From this share pool each resource provider generates a number out of a selected pair.

- 1: **for**  $i = 1$  *atleast*  $n$  **do**
- 2:   Generate a random series of pairwise relatively prime positive integers,  
 $P_i = p_{i1}, p_{i2}, \dots, p_{im}$ .
- 3:   Generate a random series of  $m$  arbitrary integers  $N_i = n_{i1}, n_{i2}, \dots, n_{im}$ .
- 4:   Place these two series  $P_i$  along with  $N_i$ , represented as  $(P_i, N_i)$  in  $SP$ .
- 5: **end for**

**Algorithm 1.2. Split Algorithm**

- 1: User  $U$  encrypts the file  $f$  with secret key  $S$ .
- 2: Split the secret key  $S$  into  $n$  number of shares,  $S_1, S_2, S_3, S_4, \dots, S_n$ .
- 3: To reconstruct  $S$  at least  $t$  number of shares are required.
- 4: **for** each share  $i$  of  $S$ , where  $i \in 1, n$  **do**
- 5:   Replicate the share  $S_i$  into  $k \geq 1$  number of replicas.
- 6:   Distribute the replicas among  $n$  number of resource providers.
- 7: **end for**
- 8: **for** each resource provider,  $R_i, i \in 1, n$  **do**
- 9:   Select a pair  $(P_i, N_i)$  from  $SP$ .
- 10:   User  $U$  saves  $(i, P_i)$  and  $R_i$  saves  $N_i$ .
- 11:   Get a unique solution  $m = x_i$  from  $(P_i, N_i)$  using Chinese Remainder Theorem (CRT).
- 12:   Split the share of secret  $S_i$  into  $m$  number of shares,  $S_{i1}, S_{i2}, S_{i3}, S_{i4}, \dots, S_{im}$ .
- 13:   Generate  $j$  number of dummy shares  $S_{ij}^D$ , where  $j \geq m$ .
- 14: **end for**

**Algorithm 1.3. Reconstruction Algorithm**

- 1: **for** each resource provider,  $R_i, i \in 1, n$  **do**
- 2:    $R_i$  asks for  $(i, P_i)$  pair from user  $U$  and  $R_i$  has  $N_i$ .
- 3:   Get a unique solution  $m = x_i$  from  $(P_i, N_i)$  using Chinese Remainder Theorem (CRT).
- 4:   Reconstruct the share of secret  $S_i$  from  $m$  number of shares,  $S_{i1}, S_{i2}, S_{i3}, S_{i4}, \dots, S_{im}$ .
- 5: **end for**
- 6: **for** each resource provider,  $R_i, i \in 1, t$  **do**
- 7:   Collect the share  $S_i$  from each resource provider.
- 8: **end for**
- 9: Reconstruct the secret  $S$  from  $S_i$  where  $i = 1, \dots, t$ .

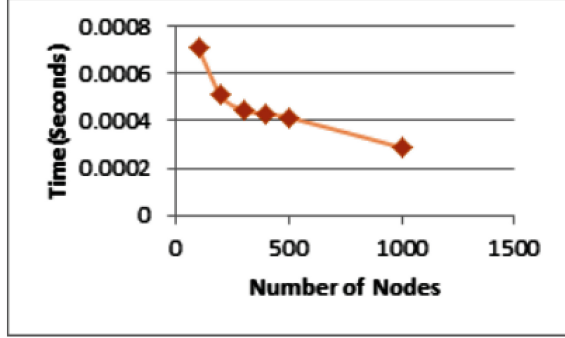
## 5 Analysis

### 5.1 Experimental Analysis

We performed simulations using a distributed cloud model that is based on the P2P overlay Kademlia [19]. We implemented our algorithm for secret sharing on the distributed cloud. We assumed that resource providers have a minimum of 2 GB RAM up to 16 GB, 2 to 8 cores. First a node identifies available resource providers near him and then divides the key into multiple shares and distributes each share to an available resource provider. The Resource provider again creates more shares by using his share as the secret and stores it. User will maintain the list of providers who store the secret shares. When a user requires the key, he contacts other resource providers who have the shares. Once resource providers receive the request, they combine the shares they have and send the actual share



to the user. Once the threshold numbers of shares are received by the user, he will reproduce the original key and use it. Figure 3 shows the average total time. The total time includes time taken to split the secret into shares at first level, find resources, distribute the shares to resource providers and split the shares at second level. We can see that as the number of nodes increases, the time to find nodes and distribute shares to nearby nodes decreases. Since the distributed cloud is formed by many users we can see our mechanism for secret sharing is feasible and efficient.

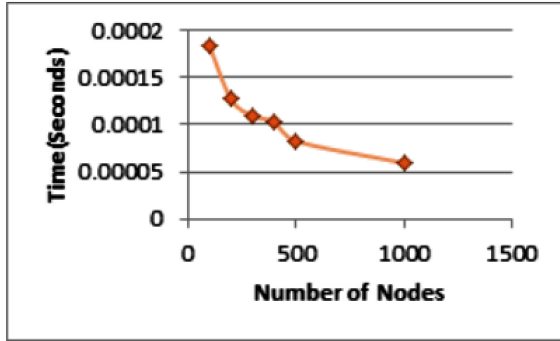


**Fig. 3.** Time to split shares and distribute them to other nodes

Figure 4 shows the time to retrieve the shares from resource providers and combine them to reproduce the secret. We see that the average time to retrieve shares from resource providers and to combine them takes significantly less time when compared to distributing the shares. In order to make sure that we have shares available to the user any time we will make replicas of shares and store them on multiple resource providers in the distributed cloud. This increases the chances of retrieving the share efficiently even when some of the nodes leave the network.

## 5.2 Security Analysis

In this paper since we are using a threshold secret sharing scheme and we are storing the shares and their replicas on multiple resource providers, compromising one or more resource providers will not disclose the secret. At the second level at each resource provider we generate equal or more dummy shares which an attacker may access as actual shares. If the resource provider is compromised by any outside attacker then the probability of using the dummy shares is greater than or equal to 0.5. This increases the chances for a resource provider to react to any security breach immediately. We create a share pool to reduce the user overhead in deciding the number of shares to be split. Since the resource provider selects  $(P_i, N_i)$  pairs randomly no eavesdropper can make a guess about



**Fig. 4.** Time to retrieve shares and combine them

the number of shares beforehand. In the distributed cloud model we have used, RP's are selected using game theory [25], which ensures that RP's are honest. This ensures that our model is not vulnerable to collusion attack

## 6 Conclusion

In this paper we propose a multilevel threshold secret sharing scheme to ensure the security of secret key in the distributed cloud environment. Our scheme can also be used for other distributed and P2P systems. Our findings show that a secret key, which plays a significant role to encrypt and decrypt the file content to be stored in cloud storage, can be stored securely in a distributed cloud environment. The increasing number of providers in the distributed cloud decreases the time to split and distribute the shares. Experimental and security analysis shows that our scheme is feasible and secure. Data replication is an other issue in distributed systems and we will look into new strategies to perform data replication to increase response time for finding data and making distributed cloud more robust.

## References

1. Shamir, A.: How to share a secret. *Commun. ACM* **22**(11), 612–613 (1979)
2. Blakley, G.R.: Safeguarding cryptographic keys. In: *International Workshop on Managing Requirements Knowledge*. IEEE Computer Society (1989)
3. Tassa, T.: Hierarchical threshold secret sharing. *J. Cryptol.* **20**(2), 237–264 (2007)
4. Beimel, A., Ben-Efraim, A., Padró, C., Tyomkin, I.: Multi-linear secret-sharing schemes. In: Lindell, Y. (ed.) *TCC 2014*. LNCS, vol. 8349, pp. 394–418. Springer, Heidelberg (2014)
5. Ito, M., Saito, A., Nishizeki, T.: Secret sharing scheme realizing general access structure. *Electron. Commun. Jpn. (Part III: Fundam. Electron. Sci.)* **72**(9), 56–64 (1989)

6. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: A New  $(k, n)$ -threshold secret sharing scheme and its extension. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 455–470. Springer, Heidelberg (2008)
7. Lin, C., Harn, L., Ye, D.: Ideal perfect multilevel threshold secret sharing scheme. In: Fifth International Conference on Information Assurance and Security, IAS 2009, vol. 2. IEEE (2009)
8. Alsolami, F., Boulton, T.E.: CloudStash: using secret-sharing scheme to secure data, not keys, in multi-clouds. In: 11th International Conference on Information Technology: New Generations, ITNG 2014. IEEE (2014)
9. Cachin, C., Haas, R., Vukolic, M.: Dependable storage in the intercloud. Research report RZ 3783 (2010)
10. Alsolami, F., Chow, C.E.: N-Cloud: improving performance and security in cloud storage. In: IEEE 14th International Conference on High Performance Switching and Routing, HPSR 2013. IEEE (2013)
11. Bessani, A., et al.: DepSky: dependable and secure storage in a cloud-of-clouds. ACM Trans. Storage (TOS) 9(4), Article No. 12 (2013)
12. Xiong, H., Zhang, X., Zhu, W., Yao, D.: CloudSeal: end-to-end content protection in cloud-based storage and delivery services. In: Rajarajan, M., Piper, F., Wang, H., Kesidis, G. (eds.) SecureComm 2011. LNICST, vol. 96, pp. 491–500. Springer, Heidelberg (2012)
13. Ding, C.: Chinese Remainder Theorem. World Scientific, Singapore (1996)
14. Anderson, D.P.: Boinc: a system for public-resource computing and storage. In: Proceedings of Fifth IEEE/ACM International Workshop on Grid Computing. IEEE (2004)
15. Anderson, D.P., et al.: SETI@ home: an experiment in public-resource computing. Commun. ACM 45(11), 56–61 (2002)
16. Khethavath, P., et al.: Introducing a distributed cloud architecture with efficient resource discovery and optimal resource allocation. In: IEEE Ninth World Congress on Services, SERVICES 2013. IEEE (2013)
17. Asmuth, C., Bloom, J.: A modular approach to key safeguarding. IEEE Trans. Inf. Theor. 30(2), 208–210 (1983)
18. Beimel, A.: Secret-sharing schemes: a survey. In: Chee, Y.M., Guo, Z., Ling, S., Shao, F., Tang, Y., Wang, H., Xing, C. (eds.) IWCC 2011. LNCS, vol. 6639, pp. 11–46. Springer, Heidelberg (2011)
19. Maymounkov, P., Mazières, D.: Kademlia: a peer-to-peer information system based on the XOR metric. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 53–65. Springer, Heidelberg (2002)
20. Chun, B., et al.: Planetlab: an overlay testbed for broad-coverage services. ACM SIGCOMM Comput. Commun. Rev. 33(3), 3–12 (2003)
21. Endo, P.T., et al.: Resource allocation for distributed cloud: concepts and research challenges. IEEE Netw. 25(4), 42–46 (2011)
22. Amazon AWS. <http://aws.amazon.com/>
23. Amazon EC2. <http://aws.amazon.com/ec2/>. Accessed on 22 July 2014
24. Microsoft Azure. <http://azure.microsoft.com>. Accessed on 22 July 2014
25. Praveen, K., Thomas, J., Liu, H.: Game theoretic approach to resource provisioning in a distributed cloud. In: International Conference on Data Science and Engineering, ICDSE 2014, pp. 51–56, 26–28 August 2014

Security in Computing and Communications

Third International Symposium, SSCC 2015, Kochi, India,

August 10-13, 2015. Proceedings

Abawajy, J.H.; Mukherjea, S.; Thampi, S.M.; Ruiz-Martínez,  
A. (Eds.)

2015, XXI, 548 p. 239 illus., Softcover

ISBN: 978-3-319-22914-0