Imporving Security For The Data Stored In Decentralized Cloud Environment Using Cloud Computing

B.NEELIMA #1, D.YASWANTH SAI #2, G.SAI BHARATH #3, K.SURENDRA REDDY #4, N.SREEKANTH #5, P.MAHESH BABU #6
[1] Asst. Professor, Department of Computer Science and engineering
[2] B.Tech., Scholars, Department of Information & Technology
QIS Institute of Technology

Abstract- Decentralized Cloud Storage is a good chance for a new cloud market, which meets an enormous community of users' supply and demand of IT resources. The dynamic and autonomous character of the resultant infrastructure presents safety issues which might slow to achieve this potential, otherwise making the projected economic advantages plainly visible and attractive. This article provides a method which will allow the owners of resources to safeguard their resources effectively and safely while depending on decentralized cloud services. Our approach combines everything-or-nothing-transform to provide robust resource protection with carefully planned techniques for resource cutting and decentralized storage network allocation. We handle both availability and security assurances, evaluate them simultaneously in our approach and allow resource owners to govern their environment.
Keywords—Distributed cloud storage. Secure destruction, splitting and distribution. Privacy.

I. INTRODUCTION

Many consumers and companies of storage/computing services from other parties have rented a clear latest trend in information technology. The previous independently managed cloud technology currently shows the participation of servers in an unknown place, wherever an Internet connection is present. The servers are often reached directly. The usage of various Internet services nowadays generally needs the presence of a service management Cloud Service Provider (CSP). The current condition is explained by several variables. Overall, IT resource procurement and management have considerable economies and major CSPs may offer services at costs lower than smaller competitors. Nevertheless, many users have an excess of computing, storage and network capacity in their own systems and would be willing to give such resources in exchange for rent. The traditional market behavior will lead to an important opportunity to create economic value from otherwise underutilized resources by using the infrastructure to support the satisfaction of supply and demand for IT services.

The rising focus on Decentralized Cloud Storage (DCS) services, typified by the availability of many nodes that may be utilized to store resources deeply decentralized, is witnesses to this transformation of terrain. Individual resources are split in sections given to various nodes (with replication to ensure availability). A resource access needs all of its shards to be retrieved. The fundamental attribute of the DSC is its co-operative and dynamic structure made up of independent nodes which can join the service and provide storage space, usually for a fee. This development was supported by Block chain-based technology, which offers an effective low-freeze electronic payment system that supports the payment of the service. Users can rent out their spare storage and bandwidth to offer a service to others in the Network, paying for that service with a networking cryptocurrency [6] on platforms like as: Storj[1], SAFE Network Vault[2], [3], IPFS[4] and Sia[5]. However, if safety concerns and the perception of (or actual) loss of control were a problem for centralized clouds and slow down, they are even more so for decentralized cloud storage, where a dynamic, independent network can indicate that owners

have a further reduction in control over where and how their resources are managed. In fact, the CSP is usually considered honest but inquisitive for centralised cloud systems and is thus relying on all the operations requested by authorized users (for example, remove a file on the owner's request) [7]. The CSP has been deterred from acting maliciously since this obviously affects its reputation. In contrast, decentralised system nodes might be behaved badly if misconduct can bring economic rewards without affecting reputation (e.g., the sale of erased file contents). Customer-side encryption generally assumed in DCSs offers a first important protective layer but, especially over the long term, it leaves resources open to attacks. For example, resources are still susceptible if an encryption key is available, or if rogue nodes do not delete the shares on request of the owner to try to rebuild the whole resource. Therefore, in DCS situations, the protection of the encryption key is not sufficient because the aforesaid dangers exist. More than one layer of defence is a common security idea. In this article, we suggest a further and orthogonal protective layer that might minimise these hazards. On the plus side, we highlight, however, that the decentralised structure of DCS systems also enhances the service's dependability, as participation by independent parties minimises the danger of the accessibility of the stored resources being limited by a single failure. The example of a mix with two encryption rounds is shown in figure 1. Contiguous mini blocks are mixed in the first round, while in the second round mini blocks are mixed to provide a mix of the entire resource content (as can be seen from the pattern coding).

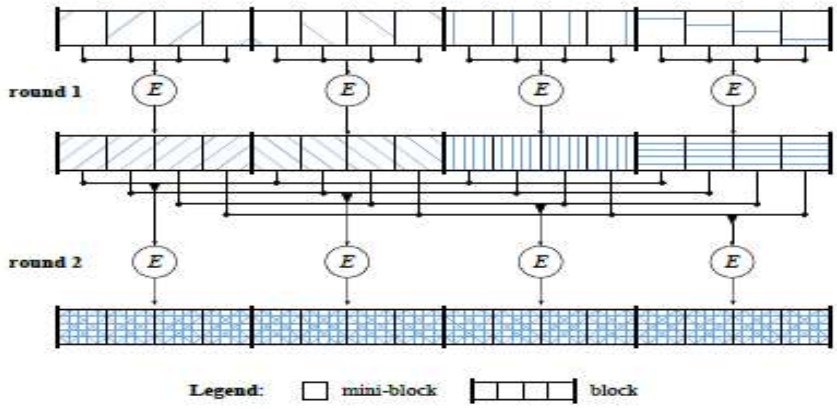

Fig. 1. An example of application of Mix & Slice

II. RELATEDWORKS

A major contribution to building dependable systems has been the RAID[12]. Normally on local discs RAID is installed. The advent of the cloud expanded RAID to take account of adversarial failures. In addition, HAIL [13] expanded RAID to include multiple cloud storage providers and a Proof of Release (PoR) [14] system in order to ensure that a provider still retains some information. However, HAIL is not appropriate for DCS systems with fewer nodes than well-established cloud providers. Furthermore, HAIL does not take into consideration the potential that adverse users try to reconstruct their own resources to benefit themselves. The solutions targeted at providing dependability and data security in DCS are the closest to ourselves. A certain level of safety assurances currently exist in several newly proposed DCS networks. (i.e., protection against all the resource slices collected by malevolent actors).

Storj[1] and Sia[5] are among them engineers of customer-side encryption and do not safeguard outsourced data from harmful coalescences. Instead, SAFE Network [2] uses a mechanism for self-encryption: a weak AONT shield is split in shares before it is uploaded. The

SAFE network's architecture and potential vectors of attacks are explored in [3]. There is a pre-determined solution in [2], [3] that does not investigate the relationship between redundeness and safety. The flexibility and security of these networks might be improved by applying our idea. The security of outsourced data can be enhanced utilising AONT as an additional line (e.g. [16], [17], [18]). However, existing methods take into account domains other than DCS. We addressed the aim of supporting policy developments in external resources when access control policy is mapped into a cryptographic policy beforehand in [9]. We discussed this idea. AONT-RS[19] is another way to use codes for AONT and Reed- Solomon.

In addition to the usage of AONT, the structure of our proposal shows a limited resemblance. In reality, the work in [19] does not take the structure of contemporary DCS systems directly into consideration and provides no strategy for identifying the parameters for the system setup. The AONT-RS development process was developed by CAONT-RS [23], which was utilised by CDStore [22] to reduce band-width and storage, and to increase resilience against lateral channel assaults, whereas DepSky [21] utilises the Shamir system to satisfy privacy needs. All of these suggestions address cloud-of-cloud setups that include cloud providers' services. Their adjustment to the DCS scenario takes considerable care and the model to identify the parameters to be used in the system setup. In addition, the relationship between safety and accessibility is not explored.

P2P systems are a predecessor to DCS. Tangler is the closely connected P2P system, which takes account of dependability and safety[23]. Tangler's aim is to defy censorship, which is not its major objective, but the possible use of DCS. In the implementation of DCS systems, some of the assumptions underlying the design of Tangler were also taken into account. One of Tangler's significant differences with our idea is the adoption of Shamir's technique, making stowage and bandwidth quite costly. Furthermore, it does not seek to combine data allocation availability and secrecy. The originality of our approach with regard to these approaches is to combine AONT with various slicing and resource allocation algorithms in DCS systems, along taking into mind security and assurances of availability. All existing systems increase their safety and availability qualities with our research of the characterization, interaction and settings of slicing and assignment guidance parameters.

III. PROPOSED SYSTEM

Fig. 2 illustrates our reference scenario. This study focuses on the design and allocation of the generated slices for the various nodes in the DCS system. Note that the term slitting in the article refers to the cutting of a resource and the term slices referring to the outcome of such a method. A slice therefore represents a piece of the resource, as opposed to a shard which represents a fraction of the resource assigned to a node (a shard can include several slices). Our focus on cutting and assignment is agnostic with regard to the exact AONT technique to be utilised, so long as the strong safeguards aimed at are guaranteed and respected.

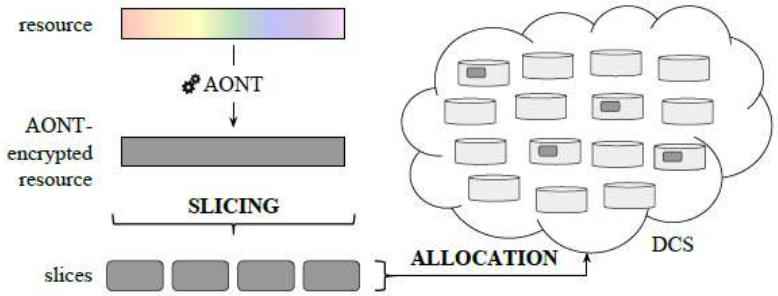

Fig 2. Reference scenario

IV. RESULTS AND DISCUSSION

We have used it in an existing DCS network to validate the benefit of our idea. We have already picked Storj as the most sophisticated and supported DCS from the existing DCS networks (e.g. Storj[1], Sia[5], IPFS[4], and Maidsafe Safe-networks[10]). The market value for these DCS (Storj for Storj, Siacoin for Sia, IPFS Filecoin, MaidSafecoin for Maidsafe)[11] confirms the importance of these solutions: the worldwide market capitalization for these efforts at the present time is more than $400 million. Currently, the Storj Network has more than 100,000 nodes, with over 100PB of data available with a 10-times-gain objective for 2019. Storj is a protocol to manage the development and enforcement of storage agreements between peers over a decentralised network. Each co-worker may negotiate contracts, upload and download data from other co-operators, and verify regularly the availability and completeness of their data. Storj uses a DHT to link interested parties to the establishment of the storage agreement. In the debate, we use the language of our concept and refer to parties that use their resources as owners for the decentralised network (in Storj). Instead, we refer to parties which give store space for a payment in a digital device as storage nodes (farmers in Storj). Bridge nodes facilitate the right system operations and might be responsible for verifying the integrity and resource availability. We explain the technical decisions, findings and some concerns regarding the consequences of fine granularity recovery, which characterise our system. The following is a description. The implementation of Min slices and Min nodes in Storj needed to modify the open source client library. Storj now offers three key clients: one C written from source, one JavaScript written and intended to work with a.js node and one Python written, compliant with any Python environment. In Python, we have also incorporated our technology for simple integration with the implementation of Mix & Slice which covers additional protective needs as well as AONT encryption (e.g., encryption-based access control and policy revocation). Storj's architecture separates the customer from the bridge and storage nodes. Our cooperation with the Python customer enables us to access the entire network of services.

We have implemented customer assignment techniques for min-slices and min- nodes and assigned slices (all slices assigned to shards in Storj's nomenclature) to nodes. The performance of shark shaping and resource rebuilding is larger in scale than the Storj network storage node performance (Mix&Slice runs at multiple hundred MB/s, whereas Storj's highest performance is around two orders lower).

We focussed our tests on assessing the time necessary to fulfil the access request as there is no substantial influence on the time requested by decryption. We note that using AONT requires that the customer can decrypt only when the full resources are available on the customer when each access request is requested. If this is an issue for the specific application domain (e.g.

very huge resources) it may be done by dividing up the large resource and using our method to the resulting (smaller) chunks. Each chunk may then be automatically downloaded and decrypted. This would lower the access time to the resources, but might also delay the completion by reducing the efficient bandwidth due to the overhead for handling of a larger number of access requests. The trials confirmed this result, which shows that the management of huge resources has a performance benefit. We have added to the client a module that switches parallel threads to (in the designated configuration, 10 concurrent threads) open requests for access to the store nodes to assess the performance of the min slices and minnodes allocation methods; In Storj, the proprietor's access requests include storage nodes and bridge nodes. Actually, every time an owner requires a shard, she requests a bridge which returns a token together with the IP address of the node, which stores the necessary shard (note that this access request is recorded and a cryptocurrency payment is created by the owner for the node). The client then uses the token as a parameter of an HTTP node request. The performance of the system in the conversation between owner and node was examined in our experimentation. In specifically, we examined, depending on the resource size, the access times recorded for both allocation techniques.

A significant constraint on Storj's present design is that shard queries are nuclear and only a certain amount of a shard controlled by a node cannot be accessed. This limitation cannot be eliminated solely by the customer, as this has a significant influence not just on storage nodes, but on the whole system structure. The Min slices and Min nodes methods have been applied accordingly. We have applied for the Min nodes concurrently. Once a node ends its shard delivery, a new request for another shard will be begun. Once the client has received full shards, the request is deemed to be complete. For Min slices, a number of t parallel threads (t = r) are activated for each shard to handle an application for separate nodes to manage the same shard, for a number of shards that are compatible with the number of concurrent threads (for ex. in experiments t + 2 and 5 Shards were being processed by 10 Threads at the same time). For Min slice, a number is activated for each shard. The group of t threads is devoted to another missing shard when a shard is entirely supplied to a customer.

Figure 4 shows the outcomes of our studies with resources of 1MB to 1GB in size used. The time needed to complete the access requests is shown in Figure 4(a) and Figure 4(b) indicates bandwidth throughput. There are three curves: one for the allocation of min slices, k = 26 and r = four and one for the assignment of min nodes with k = 12 and r = 5, respectively corresponding to the configurations shown in Figure 3(a) and Figure 3(b). The charts show the average and standard deviations of 10 executions from the values obtained. We observe that in comparison to the Min slices method the Min nodes strategy has moderate benefits. The advantage is derived from the overhead reductions associated with many nodes interaction. The inherent diversity of nodes' performance, with some being quicker than others, also adds to throughput. The approach of Min-Nodes works well, as long as the number of restricted performance nodes (slow nodes) is lower than r-1, and the performance of the shards is at least k-+1. For Min slices, it is enough to have one of the k+1 shards allocated to a number of nodes in which the contacted t nodes are sluggish to be affected by a major access delay. The method of Min Nodes works as long as the number of limited nodes (slow nodes) is lower than r - 1 and at least k + 1 node (quick node) serves as the sharp. For Min slices, it is enough to have one of the k+1 shards allocated to a number of nodes in which the contacted t nodes are sluggish to be affected by a major access delay.

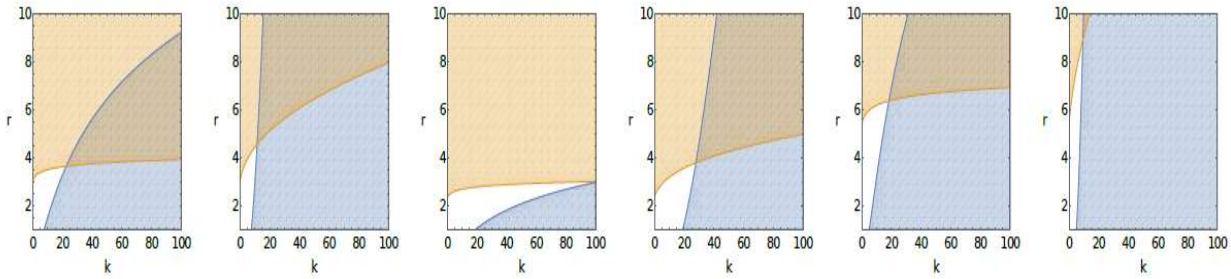

Fig. 3. Min slices and Min nodes (k; r)-allocations that guarantee Pu $\leq 10^{-7}$ and Pc $\leq 10^{-6}$ with different values for p_u and p_c

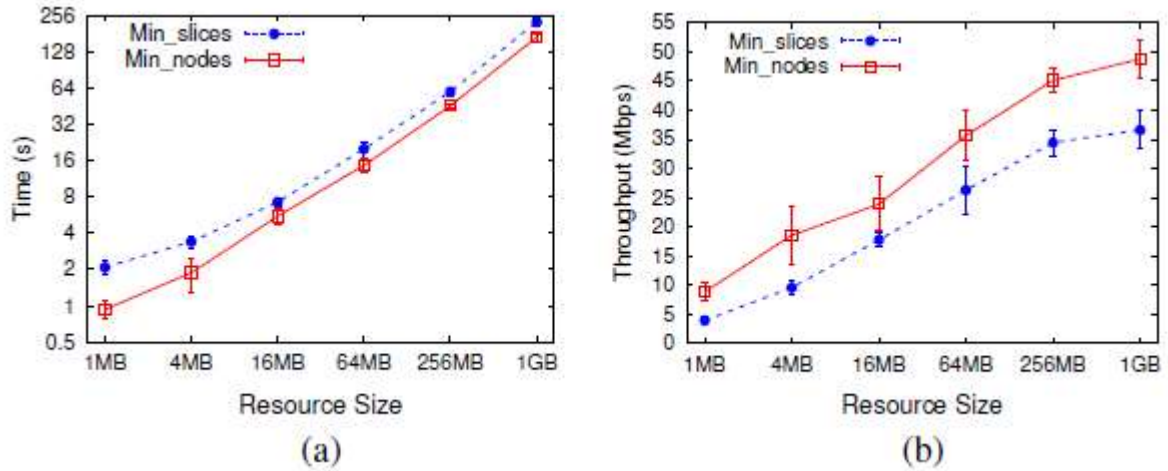

Fig. 4. Completion time (a) and overall throughput (b) in the Min slices and Min nodes allocation strategies

We saw that the existing Storj node implementation was limited by the atomic nature of the request for a shard. A system for managing partial access requests might be developed to enhance the handling of access requests substantially. This may be assumed to be a very easy modification in the server code for a generic server in a file sharing protocol; for a system of DCS this modification would require a change to the model that is used to pay access requests. The bridge should not regard every request the owner receives as an access to the entire resource (which also means payment for the entire resource), but the IP address and the authorization token of a node that stores the shard should be returned to the owner. The owner and the node should then start a communication in which the owner gives signed confirmations for resource items. This confirms can then be sent to the bridge for payment. If the owner only pays a Node for the part of the resource which is downloaded, this would lead to better performance and stronger competition among nodes; the owner would prefer the best performing Nodes and serve more traffic, which would receive the corresponding higher remuneration for their storage. Particularly for this model would be the variable structure of the Min nodes assignment. If nodes showed high access time variability, each slice could be recovered from any of the r-nodes which stored it with the ability to adjust the data transferred from each node depending on the response

time and if an r-node group were to consist of all slow nodes, the impact would be limited to single or few slices which could be recovered.

V. FUTURE SCOPE AND CONCLUSION

We have provided a technique to ensure that decentralised cloud storage services effectively safeguard resources. Our method allows resource owners to secure their resources and govern decentralised assignments to various network nodes. We have examined several techniques for the division and distribution of resources, assessing their availability and safety assurances. We have also developed issue modelling that allows owners to manage slice gritty and allocation diversification to achieve targeted safety assurances and availability. Our approach enables effective control for resource holders to remove the natural reluctance caused by safety problems and advances the implementation of new services that benefit effectively from technological development. Our approach offers opportunity for enhancements like error correction codes and techniques for information dispersion to decrease the spatial overhead.

REFERENCES

[1] S. Wilkinson, T. Boshevski, J. Brandoff, J. Prestwich, G. Hall, P. Gerbes, P. Hutchins, C. Pollard, and V. Buterin, "Storj: a peer-to-peer cloud storage network (v2.0)," https://storj.io/storjv2.pdf, Storj Labs Inc., Tech. Rep., 2016.
[2] D. Irvine, "Maidsafe distributed file system," MaidSafe, Tech. Rep., 2010.
[3] G. Paul, F. Hutchison, and J. Irvine, "Security of the maidsafe vault network," in Wireless World Research Forum Meeting 32, Marrakesh, Morocco, May 2014.
[4] J. Benet, "IPFS-content addressed, versioned, P2P file system," Protocol Labs, Tech. Rep., 2014.
[5] D. Vorick and L. Champine, "Sia: Simple decentralized storage," https: //sia.tech/sia.pdf, Nebulous Inc., Tech. Rep., 2014.
[6] C. Patterson, "Distributed content delivery and cloud storage," https: //www.smithandcrown.com/distributed-content-delivery-cloud-storage/, Smith and Crown, Tech. Rep., 2017.
[7] H. Hacig¨um¨us¸, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in Proc. of ACM SIGMOD, Madison, Wisconsin, June 2002.
[8] A. Shamir, "How to share a secret," Communications of the ACM, vol. 22, no. 11, pp. 612–613, September/December 1979.
[9] E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, and P. Samarati, "Mix&Slice: Efficient access revocation in the cloud," in Proc. of ACM CCS, Vienna, Austria, October 2016.
[10] N. Lambert and B. Bollen, "The SAFE network - a new, decentralised internet," http://docs.maidsafe.net/Whitepapers/pdf/TheSafeNetwork.pdf, MaidSafe, Tech. Rep., 2014.
[11] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3416–3452, 2018.
[12] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," ACM SIGMOD Records, vol. 17, no. 3, pp. 109–116, Jun. 1988.
[13] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A high-availability and integrity layer for cloud storage," in Proc. of ACM CCS, Chicago, IL, USA, November 2009.
[14] "Proofs of retrievability: Theory and implementation," in Proc. Of ACM CCSW, Chicago, IL, USA, November 2009.

[15] M. Albanese, S. Jajodia, R. Jhawar, and V. Piuri, “Dependable and resilient cloud computing,” in Proc. of IEEE SOSE, Oxford, UK, March 2016.
[16] A. Aldribi, I. Traore, and G. Letourneau, “Cloud slicing a new architecture for cloud security monitoring,” in Proc. of IEEE PACRIM, Victoria, Canada, August 2015.
[17] D. Nu˜nez, I. Agudo, and J. Lopez, “Delegated access for hadoop clusters in the cloud,” in Proc. of IEEE CloudCom, Singapore, December 2014.
[18] M. Theoharidou, N. Papanikolaou, S. Pearson, and D. Gritzalis, “Privacy risk, security, accountability in the cloud,” in Proc. of IEEECloudCom, Bristol, UK, December 2013.
[19] J. K. Resch and J. S. Plank, “AONT-RS: blending security and performance in dispersed storage systems,” in Proc of FAST, San Jose, CA, USA, February 2011.
[20] M. Li, C. Qin, P. P. C. Lee, and J. Li, “Convergent dispersal: Toward storage-efficient security in a cloud-of-clouds,” in Proc. of HotStorage, Philadelphia, PA, USA, June 2014.
[21] M. Li, C. Qin, and P. P. C. Lee, “CDStore: Toward reliable, secure, and cost-efficient cloud storage via convergent dispersal,” in Proc. Of USENIX ATC, Santa Clara, CA, USA, July 2015.
[22] A. Bessani, M. Correia, B. Quaresma, F. Andr´e, and P. Sousa, “DepSky: Dependable and secure storage in a cloud-of-clouds,” ACM TOS, vol. 9, no. 4, pp. 12:1–12:33, 2013.
[23] M. Waldman and D. Mazieres, “Tangler: a censorship-resistant publishing system based on document entanglements,” in Proc. of ACM CCS, Philadelphia, PA, USA, November 2001.