Efficient Share Generator for Slicing and Securely Retrieving the Cloud-Hosted Heterogeneous Multimedia Data

Khaled Riad[1,2]
[1]Department of Computer Science, College of Computer Sciences & Information Technology, King Faisal University, Al-Ahsa 31982, Saudi Arabia (e-mail: kriad@kfu.edu.sa)
[2]Mathematics Department, Faculty of Science, Zagazig University, Zagazig 44519, Egypt (e-mail: khaled.riad@science.zu.edu.eg)

Abstract

Recently, the security of heterogeneous multimedia data becomes a very critical issue, substantially with the proliferation of multimedia data and applications. Cloud computing is the hidden back-end for storing heterogeneous multimedia data. Notwithstanding that using cloud storage is indispensable, but the remote storage servers are untrusted. Therefore, one of the most critical challenges is securing multimedia data storage and retrieval from the untrusted cloud servers. This paper applies a Shamir Secrete-Sharing scheme and integrates with cloud computing to guarantee efficiency and security for sensitive multimedia data storage and retrieval. The proposed scheme can fully support the comprehensive and multilevel security control requirements for the cloud-hosted multimedia data and applications. In addition, our scheme is also based on a source transformation that provides powerful mutual interdependence in its encrypted representation—the (t, n) *Share Generator* slices and encrypts the multimedia data before sending it to the cloud storage. The extensive experimental evaluation on various configurations confirmed the effectiveness and efficiency of our scheme, which showed excellent performance and compatibility with several implementation strategies.

Keywords: Multimedia data, Cloud security, Cryptography, Shamir Secret-Sharing, Swift

1. Introduction

The proliferation of individual data commonly owns several devices, such as mobile phones, tablets, and personal computers. Also, the data generated by the smart homes, hospitals, companies, smart cities, and even military organizations is massively increasing. Multimedia data consists of various media types like text, audio, video, and animation. Accordingly, the security assurance of multimedia data is a very critical and significant issue. For example, and not as a limitation, the medical and military must contain sensitive information that unauthorized users should neither reveal nor access. Since cloud computing is the hidden back-end for storing heterogeneous multimedia data. The process of hosting multimedia data onto untrusted cloud servers is indispensable [1]. Therefore, it is urgent to support the security of multimedia data and design efficient schemes for securely storing and retrieving such sensitive data [2], [3].

Many research studies have been conducted to address the multimedia security issues, such as in [4], the authors deal with security and privacy issues for multimedia database management systems. Also, in 2018 the Multimedia Tools and Applications Journal has organized a special issue [5], which addresses a variety of security and privacy issues of multimedia big data in mobile and cloud computing and security of other aspects of mobile and cloud networks emphasizes many open questions. The authors in [6] also developed and designed a resource-efficient encryption algorithm system that applies the multithreaded programming process to encrypt the big multimedia data. However, most of the existing contributions mainly focus on document protection without considering the spatial and temporal

DOI: 10.5281/zenodo.5226865
Received: March 19, 2021 Accepted: June 12, 2021

characteristics of multimedia data. As well as, there is no consideration for the size of multimedia data to be stored or retrieved by neglecting the object size in their experiments. To the best of our knowledge, the heterogeneous multimedia data should be sliced in case of storage and retrieval for multimedia data privacy and efficiency.

We think about slicing the heterogeneous multimedia data, then encrypting these slices before sending them to the untrusted cloud servers. The data owner has to slice and encrypt its own data with a specific key. On the other hand, on the client's side, the requested multimedia data slices are only downloaded, and the data are extracted from it. There is no need to download the whole multimedia database for secure decryption. Blakley [7] and Shamir [8] introduced schemes for the case where the places whose cardinality is at least a specific threshold that is the threshold scheme.

In this scheme, we consider using (t, n) Secret sharing was introduced in [7], [8] to store multimedia data slices on the cloud servers. However, the size of the shares must not become less than the size of the secrets (multimedia data slices). Then, the amount of share which servers have been increased by more than n times as a whole. That is to say, the amount of share is represented by the multiplication of the number of shares and the size of each multimedia data. To solve this problem, we consider using the (t, ℓ, n) Ramp scheme [9], where the size of share can be reduced to $1/\ell$, compared with that when (t, n) Secret sharing is used. However, one drawback of using a Ramp scheme is that as the size of shares is reduced, it gradually becomes possible to obtain the secret from fewer than t pieces of shares. In summary, the (t, n) Secret sharing is not suitable for storing multimedia data on cloud computing because of the problem of the size of the share, and the Ramp scheme is also not ideal because of the gradual information leakage.

1.1 Motivation

In addition to reducing the amount of share by encoding multiple secrets with one key and the single key problem, a set of issues arise. This is behind our motivation to introduce our *share generator* for slicing and secure retrieval of cloud-hosted multimedia data.

- In heterogeneous multimedia data, the user who reconstructs multimedia data slices is distinct from the user who distributes them,
- The user who distributes multimedia data slices should encrypt these slices using different keys and distribute the ciphertext. Because if these slices are encrypted by the same key, once another user reconstructs a slice, he can get the decryption key for all slices, and
- If we reduce the amount of share, it should encrypt these slices in the same encryption key. Thus, the amount of share is adequate only for the multiple slices reconstructed by one user or when each slice is sufficiently more significant than the encryption key.

Therefore, there is an urgent need to introduce a scalable and distributed *share generator* scheme, which is integrated with cloud computing for securely storing and retrieving heterogeneous multimedia data.

1.2 Contributions

One of the primary contributions of this paper is incorporating the Shamir Secret-Sharing scheme [8] with cloud storage to ensure both the retrieval efficiency and secrecy of the heterogeneous multimedia data stored on the untrusted cloud servers. Our proposed scheme can encrypt the multimedia data slices before storing them on the untrusted cloud servers and reconstruct the requested multimedia data slice only, without downloading the whole multimedia database. The primary contributions can be outlined as follows:

- We have integrated Shamir Secret-Sharing scheme with cloud computing to ensure the sensitive multimedia data security when sending them to the untrusted cloud servers and during receiving it;
- We have implemented the proposed scheme on the top of our private cloud environment by initializing a set of virtual instances that are divided into three categories based on their configuration and role in the proposed scheme;
- We have analyzed the performance of our proposed scheme under three distinct situations–overlay situation, storage situation, and client situation; and
- The experimental results have confirmed the effectiveness and efficiency of our proposed scheme, which is compatible with all reasonable scenarios.

1.3 Organization

DOI: 10.5281/zenodo.5226865
Received: March 19, 2021 Accepted: June 12, 2021

This paper is organized as below. Section 2 presents the related work and discussion. The scheme preliminaries represented in the basic understanding of the threshold Shamir secret-sharing scheme are introduced in Section 3. This is followed by the problem we will solve, and the flawed strategy is presented in Section 4. Section 5 offers our proposed *t-out-of-n* scheme. The scheme implementation is presented in Section 6. Our performance analysis according to two different scenarios is introduced in Section 7. This is followed by the conclusions in Section 8. v

2. Related Work

The notion of producing the extraction of the information content of an information origin must think about the security of the decryption process. Also, downloading only the requested slice of information instead of downloading the whole object. There is a great growth in multimedia data hosted on cloud computing. Dealing with multimedia data through cloud computing is a challenge [10]. The storage and retrieval of multimedia data while linking it to either cloud computing or big data is a hot topic. As well, content-based multimedia encryption and deduplication are new and open areas. The authors in [11] have proposed a secure tensor singular value decomposition (S-tSVD). They have introduced various multimedia data as cipher subtensors, using fully homomorphic encryption. They have introduced a new multiplication operation based on the Fourier transform. The authors in [12] have investigated access control and authorization functions, and they have discussed how to advance them for network slicing deployments with continuous and closed-loop usage control mechanisms.

Since storing small amounts of multimedia data leads to a significant standard deviation of storage space, a high failure rate of storage nodes, and poor quality of data storage. The authors in [13] have introduced a machine learning-based multimedia key area synchronous storage method. The method uses a genetic algorithm to calculate the distance between data in multimedia critical area and cluster center, and redistribute cluster set, K-Mean is realized by M-R parallel computing model. To meet the needs of multimedia communication in the multi-cloud environment and improve the experience quality of mobile multimedia users. The authors in [14] proposed an autonomous computer system based on cooperative cognition and multimedia data behavior measure. The authors in [15] exploited textual reviews and item images together with ratings to tackle the latent factor model limitations. However, it suffers from many problems, including cold-start, non-transparency, and suboptimal results for individual user-item pairs limitations. To improve the storage efficiency and reduce the management expenditure of these massive multimedia data in heterogeneous networks, the direct way is multimedia data deduplication. However, it raises serious privacy concerns and poses new security challenges, such as privacy leakage, side-channel attacks, and unauthorized access. The authors in [16] introduced an adaptive access control based on XACML access policies used for heterogeneous distributed IoT environments.

The multimedia data security and privacy are always hot topics. The authors in [17] propose a novel role symmetric encryption (RSE) algorithm and an RSE-based proof of ownership (RSE-PoW) scheme for secure deduplication in hierarchical heterogeneous environments. The authors in [18] have introduced a new operative scheme called RoughDorid for detecting malware applications directly on the smartphone. The authors in [19] suggested a novel data retrieval scheme for multiple users based on the lattice-based mechanism. There is another contribution [20] for applying access control in the cloud by introducing the trust notation for granting or denying access, which can be extended and used for the multimedia data. The authors in [21] proposed a scheme that approximates the Sigmoid function as a polynomial function to support the secure computation of the activation function with the BGV encryption. Only the client performs encryption and decryption operations in our scheme, while all the computation tasks are performed on the cloud. But in our scheme, the encryption and decryption are executed by using the *share generator* that is secure and efficient in doing that. In this way, there is no computation overhead on the owner or the user. The authors in [22] have developed a novel aspect-aware recommender model named A3NCF, which can capture the varying aspect attentions that a user pays to different items. Finally, the authors in [23] introduced a dynamic access control for IoT devices using multi-authority cloud storage.

According to multimedia data retrieval, the authors in [24] presented a practical and economic architecture named SHMR (Semantic-based Heterogeneous Multimedia Retrieval), which uses low cost to store and retrieve semantic information from heterogeneous multimedia data. Also, the authors in [25] proposed Multi-graph Cross-modal Hashing (MGCMH) based on unsupervised multi-graph cross-modal hashing. The proposed model comprehensively considers three fundamental problems that should be seriously considered in multimedia retrieval. The authors in [26]

DOI: 10.5281/zenodo.5226865
Received: March 19, 2021 Accepted: June 12, 2021

have used private information retrieval to guarantee secure IoT data storage and retrieval from untrusted cloud servers. The authors in [27] presented an optimized algorithm called semantic ontology retrieval (SOR), which uses big data processing tools to store and retrieve ontologies from heterogeneous multimedia data. The authors in [28] proposed a multi-view discriminative and structured dictionary learning with group sparsity and graph model (GM-GS- DSDL) to fuse different views and recognize human actions. The authors in [29] presented a novel multimedia hashing framework called Label Preserving Multimedia Hashing (LPMH) for multimedia similarity search. In LPMH, a general optimization method learns the joint binary codes of multiple media types by explicitly preserving semantic label information. The authors in [30] proposed a standard graph regularization-based modality-dependent cross-media retrieval approach (JGRMDCR), which considers the one-to-one correspondence between different modal data pairs, the inter-modality similarities, and the intra- modality similarities.

To the best of our knowledge and understanding, dealing with such numerous and massive multimedia data as a single block of data is impossible. Therefore, there should exist some schemes for segmenting these heterogeneous data to efficiently and securely manage it. Our proposed scheme slice, encrypt, and then host the multimedia data on the untrusted cloud servers, to benefit from the cloud computing storage advantages. At the same time, we can securely retrieve only the requested multimedia data slices without downloading the whole database first for secure decryption.

3. Scheme Preliminaries

To continue with the full description of our scheme, we should first introduce one of the most essential parts in constructing our scheme, which is Shamir Threshold Secret- Sharing scheme. In the threshold secret-sharing scheme [31], the users that are allowed to reconstruct the share are the set of all users, where the size of such a group is more significant than a specific threshold. This will introduce a *t out-of n* access structure. Finally, it should be mentioned that the simplest way of reconstructing (recovering) the original data from a set of shares is shown in Figure 1.

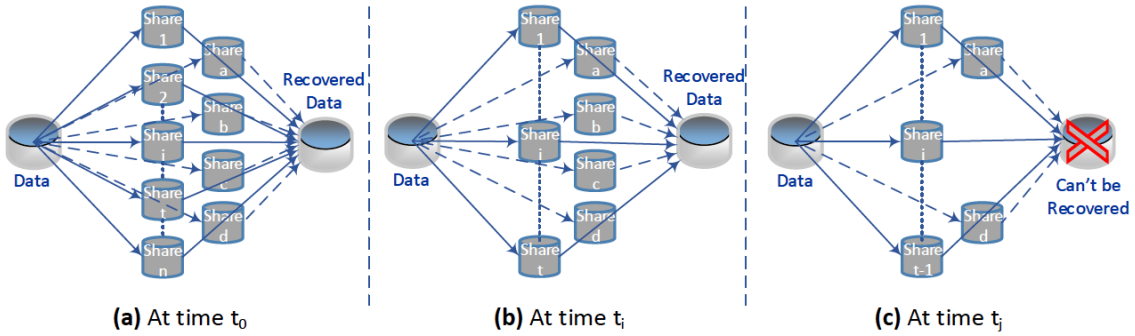

Figure 1. Shamir's threshold secret-sharing for data sharing and reconstruction.

As shown in Figure 1, we can generate n shares of the data according to the threshold secret sharing scheme, as shown in **(a)**; these n shares are distributed into n parties. At any time, we can recover the original data by collecting arbitrary t or more shares, as shown in **(b)**. With the lapse of time, for some reason, either the leaving of the shareholder or the share's losing and deletion could lead to the decline in the number of shares. When the number of shares that can be gathered is more minor than *t*, the data cannot be reconstructed, as shown in **(C)**.

4. Problem Formulation

One of our primary goals is to reduce the amount of share, which can be realized by encoding multiple secrets with one key. On the other side, the one key problem arises. Also, according to the nature of multimedia data, the user that distributes the multimedia data (data owner) is different from the user that requests access. Thus, the data owner should handle with much care the keys used for distributing his multimedia data. Therefore, the multimedia data owner should be online all the time that is not realistic. Finally, we urgently need to find an effective and successful access control to manage such situations in the cloud-hosted multimedia data.

We will introduce a novel scheme based on Shamir Secret-Sharing and prepare it to be compatible with hosting and receiving multimedia data from cloud computing environments. The proposed scheme consists of instances, devices,

DOI: 10.5281/zenodo.5226865
Received: March 19, 2021 Accepted: June 12, 2021

and machines, each of which has its configuration. Moreover, each instance, device, or machine in the system can manage a particular part of the proposed scheme, as shown in Figure 2.

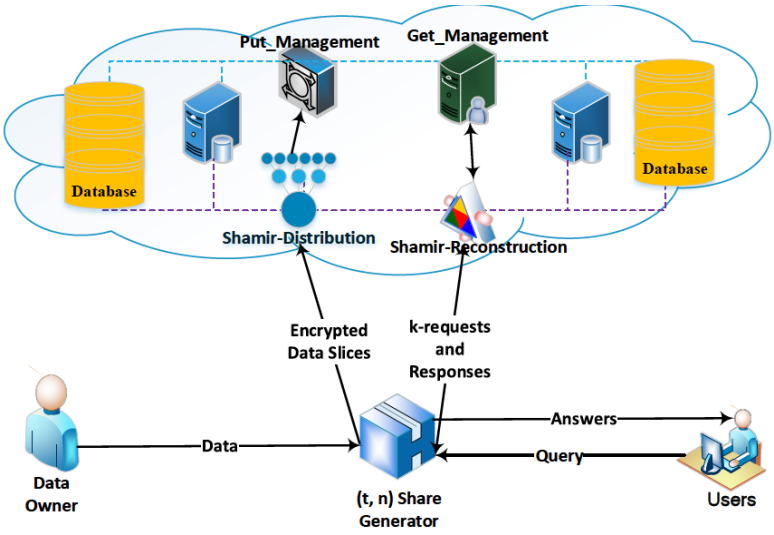

Figure 2. Cloud-integrated Shamir Secret-Sharing and *share generator* system model.

4.1 Share Generator

The (t, n) *Share Generator* in the proposed scheme is an important machine that has three different phases of communications. First, it accepts the users' queries and responds to them with the answers after receiving them. Also, it receives the multimedia data from the owners and prepares it to be stored on the cloud servers. The primary tasks of (t, n) *Share Generator* are as follows:

- Receiving multimedia data from the data owners, slicing the received multimedia data, encrypting each slice, and submitting these slices to *Shamir-Distribution* instance;
- Accepting users' queries, partitioning the query, and submitting the query set to *Shamir-Reconstruction* instance;
- Receiving the responses to the requested queries from
- *Shamir-Reconstruction* instance; and
- Submitting the answer to the user after grouping it to the user represents the answer of the submitted query by the user previously submitted to the (t, n) *Share Generator*.

4.2 Distribution and Put_Management

In this part, we have two primary virtual instances (*Shamir- Distribution*, and *Put_Management*) hosted on the cloud environment. Those instances are of X-large configuration (8 VCPUS, 160 GB disk space, and 16384 MB of RAM) to handle the massive stream of requests and processes. At first, *Shamir-Distribution* instance has the following responsibilities:

- Receive the encrypted multimedia data slices from the (t, n) *Share Generator*;
- Run *Shamir-Distribution* (Algorithm 1) to generate a set $s_1, \ldots, s_u$ of shares for each participant (multimedia data slice); and
- Submit the generated set of shares directly to the *Put_Management* instance to be stored on the cloud servers.
- The *Put_Management* instance is responsible for:
- Receiving the set of shares generated by *Shamir-Distribution* algorithm;
- Communicating with a group of cloud storage servers and database servers to store the acquired shares; and
- Storing each share on a specific server to disclose the identity of the stored encrypted shares neither in the storage process nor in requesting a particular slice of multimedia data.

DOI: 10.5281/zenodo.5226865
Received: March 19, 2021 Accepted: June 12, 2021

It needs to be mentioned here that *Shamir-Distribution* and *Put_Management* instances are on the same network of the cloud servers and database servers. Thus, the communication between them has no time.

4.3 Reconstruction and Get_Management

We can find on the other side from the Distribution and *Put_Management*, reconstructing the previously-stored shares is done based on two practical instances (*Shamir-Reconstruction* and *Get_Management*). Once *Shamir-Reconstruction* receives k-requests from the (t, n) *Share Generator* will forward the requests to the *Get_Management* instance. As one of the essential duties of *Get_Management* instance, it will get k-answers from those servers hosting these answers. The k-answers are the direct responses for the k-requests by *Shamir-Reconstruction*. The *Get_ Management* instance has the following duties:

- Get k-answers from multiple cloud storage serves that are the answers to the k-requests introduced by *Shamir-Reconstruction* instance; and
- Send the gotten k-answers to another algorithm (*Shamir-Reconstruction*).

The *Shamir-Reconstruction* instance has a critical role:

- Receive k-request from the (t, n) *Share Generator* machine;
- Receive the k-answers from the *Get_Management* instance;
- Run *Shamir-Reconstruction* (Algorithm 2) to generate the secret S of a specific share requested by the (t, n) *Share Generator*; and
- Send the generated share to the requesting *Share Generator*.

It should be mentioned that *Shamir-Reconstruction* and *Get_Management* instances are of X-large configuration.

4.4 Cloud Storage

In our scheme, most of the considered instances and machines are implemented based on a private cloud environment built using OpenStack [32]. The cloud environment is built using three physical servers with high physical capabilities working as controller, neutron, and compute nodes. We have initialized various virtual instances with Large (4 VCPUS, 80 GB disk space, and 8192 MB of RAM) configurations on the same local network working as cloud storage servers. As well as initializing some virtual instances with X-large configuration, working as database servers. It should be mentioned that both the cloud storage servers and the database servers are on the same network.

The cloud storage and database servers are organized into multiple tenants; each tenant has its tenant manager. This organization eases the work of both the *Put_Management* and *Get_Management* servers in storing and reconstructing its shares. The tenant manager is responsible for storing the shares generated by *Shamir-Reconstruction* and provided to it by the *Put_Management* instance. Also, the tenant manager is responsible for getting the shares back from the servers under its management and send them back to the *Shamir-Reconstruction* through the *Get_Management* instance.

4.5 Users and Data Owners

The multimedia data owners care about the security of their multimedia data, and they are always worried that the cloud servers may introduce their data to unauthorized users. Therefore, the data owners provide the multimedia data to the (t, n) *Share Generator* that will slice and encrypt it before submitting it to the cloud storage servers. On the user's side, the user has to issue a query that will be verified by the (t, n) *Share Generator* before forwarding it to *Shamir-Reconstruction*.

5. Our Construction

Our proposed scheme integrates the Shamir Secret-Sharing scheme with cloud computing to guarantee private information security during storing and receiving it from the untrusted cloud servers. Our scheme consists of three primary stages: *Encryption and Storage, Authorization and Query, and Decryption and Reconstruction*.

5.1 Encryption and Storage

The multimedia data $\mathcal{D}$ is going to be stored on multiple untrusted cloud servers using (t, n) Secret sharing. In cloud storage, we believe the users who reconstruct the multimedia data are different from those who spread (owner) it. In

DOI: 10.5281/zenodo.5226865
Received: March 19, 2021 Accepted: June 12, 2021

cases like this, the owner should encrypt its multimedia data using other keys and distribute the ciphertext in the same way as in [33]. He can find the key because in the event precisely, the same key encrypts the information shares reconstructs the information. Thus, we will consider two managing servers to store multimedia data, as shown in Figure 2. The *Put_Management*, which receives the encrypted multimedia data slices and stores them on the untrusted storage servers under its control, and the *Get_Management*, which manages the requests from the (t, n) *Share Generator*.

Definition 1: (*$(\boldsymbol{t, n})$ *Shamir Secret-Sharing): (t, n)-threshold secret sharing scheme is a method of distributing a secret to n participants in such a way that any group of t (threshold) or more participants can later find the secret $(1 \leq t \leq n)$, whereas no group of fewer than t participants can do so. Each participant is given some partial information, called a share. It consists of two algorithms:

- **Distribute:** The distribution algorithm carries the key S and divides S to P shares, with every P_i getting a share s_i (Algorithm 1).
- **Reconstruct:** The reconstruction algorithm gathers every share s_i from every P_i and utilizes the shares to reconstruct the key S (Algorithm 2).

Algorithm 1: Shamir-Distribution
Input: Secret $S \in \mathbb{F}_q$
Input: A set $P = P_1, \dots, P_u$ of participants
Input: A threshold value $1 \leq t \leq n$
1: Select random values $r_1, \dots, r_{u-1} \in \mathbb{Z}_p^R$
2: Construct the polynomial $r(x) = s + r_1 x + \cdots + r_{u-1} x^{u-1}$
3: **for** $i = 1$ to n **do**
4: Send share $s_i = r_i$ to P_i
5: **end for**
Output: A set $\{s_1, \dots, s_u\}$ of shares for each p_i

Algorithm 2: Shamir-Reconstruction
Input: A set $X \subset P_1, \dots, P_u$ of size u participants
Input: A set of shares s_i for each $x_i \in X$
1: Compute $S = \sum_{x_i \in X} s_i \lambda_{X,x_i}$ with $\lambda_{X,x_i} = \prod_{j \in Xi} \frac{j}{j-1}$
Output: The secret S

The secret sharing scheme will be employed to store multimedia data $\mathcal{D}$, which will be split into $(x_1, x_2, \dots, x_j, \dots, x_n)$. Our scheme will encrypt $(x_1, x_2, \dots, x_j, \dots, x_n)$ as stated by the access structure $\mathcal{A}$. We will presume that the ciphertext implicitly includes the access structure. Then, we will utilize the *Hash function* as a permutation function, which is used for hiding the index of the portioned slices of multimedia data to be stored. Our storage objective is to get each component stored on each server for each piece of the multimedia data to be stored $\left(Share(\mathcal{D}, k, t, x_i) = \left[S_1^i, S_2^i, \dots, S_k^i\right]\right)$.

The proprietor encrypts the multimedia data pieces with a content key, each using symmetric encryption procedures. It then encrypts the content essentially. It requires as input the public primary key managed by Shamir secret sharing scheme. The encryption process is illustrated as follows: There is a content key ck and an access structure $\mathcal{A} = (M, \rho)$. The access structure will first encrypt the multimedia data, and the content key is also used in the encryption process before sending the multimedia data to the untrusted cloud servers. Also, the encryption process takes as input the multimedia data Public Key (*PK*), which Shamir Secret-Sharing manages. Then, we should choose random secrets $\left(\alpha_i, \beta_i, \gamma_i \in \mathbb{Z}_q^*\right)$ for each multimedia data slice x_i to be encrypted. Also, choose a random encryption exponent $f \in \mathbb{Z}_q^*$.

DOI: 10.5281/zenodo.5226865
Received: March 19, 2021 Accepted: June 12, 2021

After that, the encryption process chooses a random vector $\vec{v} = [f, y_2, \dots, y_n]^T \in \mathbb{Z}_q^{*n}$ that is used to share the random encryption exponent f. The encryption process will compute $\lambda_j = \vec{v} \,.\, M_j$ for each *j* ranging from *1* to ℓ. The process then randomly chooses $r_1, \dots, r_l \in \mathbb{Z}_q^*$. After that, compute four basic components that are used in composing the ciphertext,

$$C' = g^f, \tag{1}$$

$$C'' = g^{\frac{f}{\beta_i}}, \tag{2}$$

$$C_j = g^{a\lambda_j}.((g^{\vec{v}(j)\rho(j)}H(\rho(j))^{\gamma_i})^{-r_j}, \tag{3}$$

$$D_{1,j} = g^{\frac{r_j}{\beta_i}}, \tag{4}$$

$$D_{2,j} = g^{-\frac{\gamma_i}{\beta_i}r_i}. \tag{5}$$

Finally, compute the ciphertext by using the previously computed components and the content key,

$$CT_i = \left[ck_i.\left(\prod_j^l \hat{e}(g,g)^{\alpha_i}\right)^f, C', C'', C_j, D_{1,j}, D_{2,j}, \rho(j)\right] \tag{6}$$

5.2 Authorization and Query

Once the user has been authorized to the system, as one of its rights, he can issue a request to receive a specific piece of information stored on the untrusted cloud servers. There has to be a token for every user since the data is hosted on both cloud servers. Then, the user has to issue a query to the (t, n) *Share Generator,* including the user's secret key, attributes, and certificates. The (t, n) *Share Generator* will issue *k*-queries (one for each server) $\mathcal{R} = \{r_1, r_2, \dots, r_i, \dots, r_k\}$for the get management point, where $r_i = \mathcal{R}_i(k, n, j)$ and j is randomly chosen by flipping coins. Every server will reply with one encrypted reaction, the consumer will have an encrypted Response Collection $RC = \{rc_1, rc_2, \dots, rc_i, \dots, rc_k\}$, where $rc_i = RC_i(k, i, x, r_i)$. The get management point will respond with all the servers' response collection, sent to the user requesting access.

5.3 Data Decryption

When the user requesting access receives the response collection RC, which implicitly contains the access structure $\mathcal{A}$. Based on the user's secret key USK, and the user's set of attributes USA. When the USA suits the access structure, the consumer can get the response R collection, which will be utilized for the decryption. In our scheme, the secret sharing scheme has a variable reconstruction threshold (t).

6. Implementation

We have simulated our proposed scheme using our private cloud environment built based on OpenStack [32]. The construction of the proposed scheme is mainly based on the initialization of a reasonable collection of virtual instances with different configurations based on their role in the proposed scheme.

We have three groups of virtual instances:

- **Group I:** A number of virtual instances are working as untrusted cloud storage servers and database servers. Those instances are grouped into the same Internet Protocol address scope with the same subnet mask, and they have excellent storage capabilities;
- **Group II:** Three instances with medium configuration. Those instances are working as a *Share Generator* server, *Shamir-Distribution* server, and *Shamir-Reconstruction* server; and
- **Group III:** Two virtual instances with extensive configuration. Those instances are working as Put Management point and get management point.

7. Performance Analysis

In this section, we will analyze the performance of our proposed scheme under three distinct situations–overlay situation, storage situation, and client situation.

DOI: 10.5281/zenodo.5226865
Received: March 19, 2021 Accepted: June 12, 2021

7.1 Overlay Situation

For the purpose of our scheme, the Overlay solution is analyzed using as a reference the Swift service. Swift has been selected due to its popularity, availability as open-source, and technical features that are good representatives of what is offered by a modern object storage service for the cloud. We consider two main alternatives for realizing our scheme on Swift1 without any changes to the server. The first option assumes to manage each fragment as a separate object. The second option uses the ability to access portions of objects and specifically considers Dynamic Large Objects (DLOs). Our experiments show that this latter option provides significant benefits in performance with respect to managing fragments as separate objects. DLOs deserve then to be used when available.

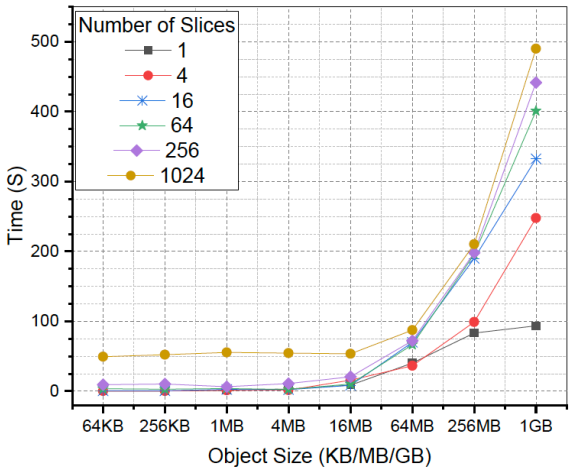

Figure 3. The time overhead for executing multiple get requests, according to the size of objects on Swift.

Figure 3 compares the time required for the execution of get requests for different numbers of slices with the assumption that each slice is related to a specific object size. The lines correspond to distinct values for the number s of slices (i.e., 1, 4, 16, 64, 256, and 1024). The parameters that drive the performance are the network bandwidth and the overhead imposed by the management of each request. Forget requests, the overhead introduced by managing one request for each slice dominates when the resource is small, whereas the increase in object size makes the network bandwidth the bottleneck. The profile of put requests uploading the complete resource proved identical to the profile of get requests using a single slice. The execution of put slice requests grows linearly with the size of the slice.

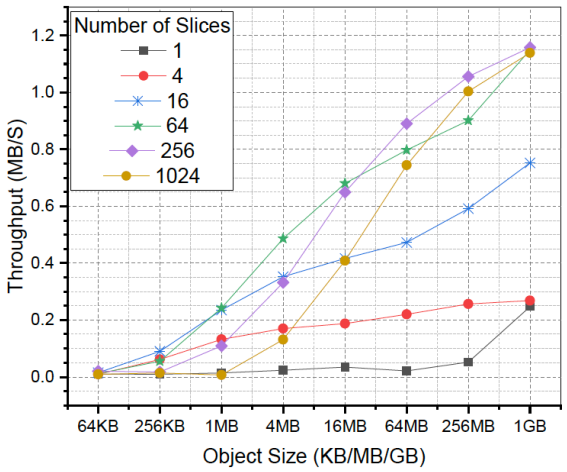

Figure 4. The throughput for a workload of executing multiple get and put slice requests on Swift, according to the size of objects.

DOI: 10.5281/zenodo.5226865
Received: March 19, 2021 Accepted: June 12, 2021

The identification of the best number of slices is not an easy task. It requires considering the profile of the scenario. We evaluated the behavior of a system on a collection of 1000 objects where, after each put slice request, a sequence of 50 get requests were executed on objects in the collection, all of the same size. Figure 4 reports the results of the best number of slices experiments. As objects become larger, the benefits of slicing in the application of policy updates compensate for the overhead imposed on retrieving the objects. It is to note that the solution's performance that does not use our scheme corresponds to the line with one fragment. The throughput of the configurations using slices is orders of magnitude higher already for medium-size objects. The graph also shows that the best number of slices depends on the resource size. The identification of the value to use requires considering the configuration of the system and the expected workload.

The customer only creates one get request for the item, separately from the number of items. The object's descriptor can be prolonged use the representation of every fragment's edition. We have embraced the DLO service provided by Swift to execute the get and also put slice methods. Since the storage service is based on high capabilities cloud servers, we do not care about the storage overhead itself. But we are laying more importance on the part related to replying to the user with the result of a specific request.

We are interested in the get slice request execution time, as shown in Figure 3. We are considering six different slice sizes (1, 4, 16, 64, 256, and 1024), and the results are the average of 25 trials at the same environmental conditions. We have found that at a 1 GB size object, the time required to process the request internally is about 92 seconds for one considered fragment; the time is 1000 seconds for 1024 supposed fragments for the same object size.

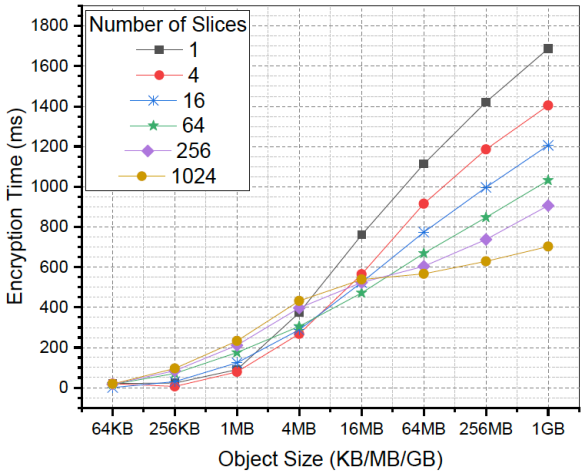

Figure 5. The encryption time in milliseconds for encrypting multiple slices of multimedia data according to the size of objects.

7.2 Storage Situation

In our scheme, the primary storage overhead is induced by the encryption algorithm. At first, the data owner sends the multimedia data to be stored for the *share generator*. Then, the shares (slices) are sent to *Shamir-Distribution*, and after that for the *Put_Management*.

Figure 5 introduces the encryption time elapsed at the *Share Generator* for encrypting multiple slices for different object sizes. The lines correspond to distinct values for the number s of slices (i.e., 1, 4, 16, 64, 256, and 1024). The encryption time elapsed for encrypting one request for each slice dominates when the resource is small, whereas the increase in object size makes the encryption algorithm runs for a long time to encrypt it successfully. In comparison, when considering 1024 slices for a 1GB multimedia data object, the encryption time is small compared to considering only one slice for encrypting it. This is due to the fact that encrypting multiple small slices is faster than encrypting few large objects. It should be mentioned that the encryption time of our scheme is better than the time for the traditional encryption schemes. It should be noted that the experimental results for that part are an average of 25 trials.

DOI: 10.5281/zenodo.5226865
Received: March 19, 2021 Accepted: June 12, 2021

7.3 Client Situation

In our scheme, the user is requested to execute a bit complex decryption process. Compared with the use of the traditional Attribute Encryption Scheme (AES). Based on our great VCPUs, the decryption stage for a specific slice of data is executed parallelly. Then, the user can slightly decrypt the received RC in a minimal time compared to the traditional AES scheme. In our experiments, we have considered four different size categories: 32:127 bits; 128:511 bits; 512:2047 bits; and 2048:8192 bits.

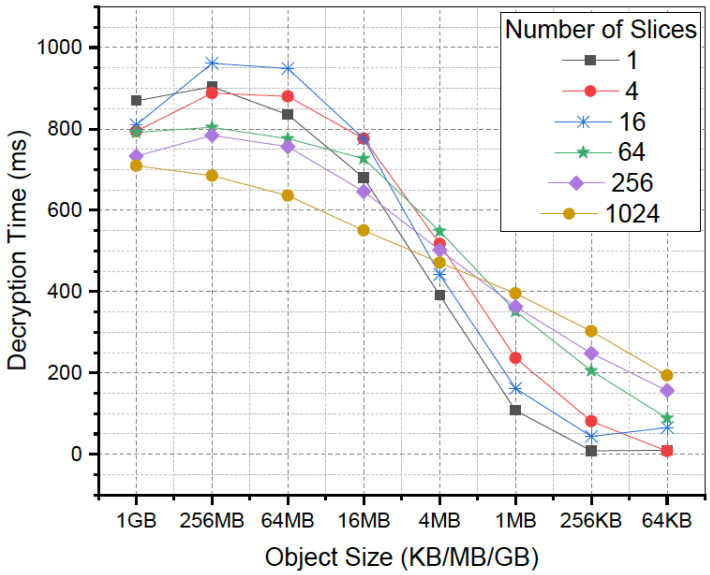

Figure 6. The client-side decryption time in milliseconds for decrypting multiple multimedia data slices according to the object size.

Figure 6 shows the decryption time at the client-side. It is clear that the decryption time is decaying with decreasing the object size. Also, when requesting multiple slices, the decryption time decay. This is due to the fact that decrypting multiple small slices do not cause network congestion. On the other hand, few massive objects cause significant network overhead and congestion. Thus, from this figure, the most oversized objects can be decrypted very vastly when considering a vast number of slices for slicing them. It should be noted that the experimental results for that part are an average of 25 trials.

8. Conclusions

The security and privacy of multimedia data are critical and challenging, especially with the massive amount of multimedia data and applications. Also, with the enormous need for cloud computing to be the back-end for storing heterogeneous multimedia data. Moreover, numerous security issues arising when hosting sensitive data on cloud untrusted servers. Therefore, there must exist an effective and secure scheme for efficiently storing and retrieving the multimedia data from the untrusted cloud servers without revealing the identity of the portion being retrieved. This paper discusses securely storing and retrieving the multimedia data from the untrusted cloud servers without revealing the identity of the portion being retrieved. Accordingly, we have proposed a scheme for efficiently and securely storing and retrieving the heterogeneous multimedia data from the untrusted cloud servers. This scheme slices the multimedia data, encrypts them and stores them on the cloud servers. The requested slice of data can only be downloaded on the client-side without downloading the whole multimedia data object. Then, it can be decrypted if the user is authorized for the decryption process. Therefore, our scheme guarantees both the storage security and retrieval of multimedia data. Three distinct situations–overlay situation, storage situation, and client situation are used to evaluate our proposed scheme and demonstrate its effectiveness and practicality in security modeling for multimedia storage and retrieval systems.

DOI: 10.5281/zenodo.5226865
Received: March 19, 2021 Accepted: June 12, 2021

References

[1] D. Quick and K.-K. R. Choo, "Big forensic data management in heterogeneous distributed systems: Quick analysis of multimedia forensic data," Softw. Pract. Exper., vol. 47, no. 8, pp. 1095–1109, Aug. 2017.

[2] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K. K. R. Choo, "Fuzzy identity- based data integrity auditing for reliable cloud storage systems," IEEE Transactions on Dependable and Secure Computing, pp. 1–1, 2017.

[3] Z. Liu, K.-K. R. Choo, and M. Zhao, "Practical-oriented protocols for privacy-preserving outsourced big data analysis: Challenges and future research directions," Computers & Security, vol. 69, pp. 97 – 113, 2017, security Data Science and Cyber Threat Management.

[4] B. Thuraisingham, "Security and privacy for multimedia database management systems," Multimedia Tools and Applications, vol. 33, no. 1, pp. 13–29, Apr 2007.

[5] B. B. Gupta, S. Yamaguchi, and D. P. Agrawal, "Advances in security and privacy of multimedia big data in mobile and cloud computing," Multimedia Tools and Applications, vol. 77, no. 7, pp. 9203–9208, Apr 2018.

[6] S. Aljawarneh, M. B. Yassein, and W. A. Talafha, "A multithreaded programming approach for multimedia big data: encryption system," Multimedia Tools and Applications, vol. 77, no. 9, pp. 10 997–11 016, May 2018.

[7] G. R. Blakley, "Safeguarding Cryptographic Keys," in AFIPS, International Workshop on, Managing Requirements Knowledge, vol. 48, 1979, pp. 313–317.

[8] A. Shamir, "How to Share a Secret," Communications of the ACM, vol. 22, no. 11, pp. 612–613, 1979.

[9] G. R. Blakley and C. Meadows, Advances in Cryptology, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, vol. 196, no. 0302-9743, ch. Security of Ramp Schemes, pp. 242–268.

[10] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan], "The rise of "big data" on cloud computing: Review and open research issues," Information Systems, vol. 47, pp. 98 – 115, 2015.

[11] C. Fu, Z. Yang, X. Liu, J. Yang, A. Walid, and L. T. Yang, "Secure tensor decomposition for heterogeneous multimedia data in cloud computing," IEEE Transactions on Computational Social Systems, vol. 7, no. 1, pp. 247–260, 2020.

[12] B. Martini, P. Mori, F. Marino, A. Saracino, A. Lunardelli, A. L. Marra, F. Martinelli, and P. Castoldi, "Pushing forward security in network slicing by leveraging continuous usage control," IEEE Communications Magazine, vol. 58, no. 7, pp. 65–71, 2020.

[13] L. Chen, "Analysis of synchronized storage method for multimedia key areas based on machine learning," Multimedia Tools and Applications, May 2019.

[14] Y. Xiao, L. Zhang, and L. Hou, "Autonomous multimedia cluster computing based on cooperative cognition data behavior measurement under multi-cloud computing," Multimedia Tools and Applications, vol. 78, no. 7, pp. 8783–8797, Apr 2019.

[15] Z. Cheng, X. Chang, L. Zhu, R. C. Kanjirathinkal, and M. Kankanhalli, "Mmalfm: Explainable recommendation by leveraging reviews and images," ACM Trans. Inf. Syst., vol. 37, no. 2, pp. 16:1–16:28, Jan. 2019.

[16] K. Riad and J. Cheng, "Adaptive XACML access policies for heterogeneous distributed IoT environments," Information Sciences, vol. 548, pp. 135-152, 2021.

[17] J. Xiong, Y. Zhang, X. Li, M. Lin, Z. Yao, and G. Liu, "Rse-pow: a role symmetric encryption pow scheme with authorized deduplication for multimedia data," Mobile Networks and Applications, vol. 23, no. 3, pp. 650–663, Jun 2018.

[18] K. Riad and L. Ke, "Roughdroid: Operative scheme for functional android malware detection," Wireless Communications and Mobile Computing, vol. 2018, pp. 1–10, 2018.

[19] Y. Yang, X. Zheng, V. Chang, S. Ye, and C. Tang, "Lattice assumption based fuzzy information retrieval scheme support multi-user for secure multimedia cloud," Multimedia Tools and Applications, vol. 77, no. 8, pp. 9927–9941, Apr 2018.

[20] K. Riad and Z. Yan, "Multi-Factor Synthesis Decision-Making for Trust-Based Access Control on Cloud," International Journal of Cooperative Information Systems, vol. 26, no. 4, pp. 1–33, 2017.

[21] Q. Zhang, L. T. Yang, and Z. Chen, "Privacy preserving deep computation model on cloud for big data feature learning," IEEE Transactions on Computers, vol. 65, no. 5, pp. 1351–1362, May 2016.

[22] Z. Cheng, Y. Ding, X. He, L. Zhu, X. Song, and M. Kankanhalli, "A3ncf: An adaptive aspect attention model for rating prediction," in Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 3748–3754.

[23] K. Riad, T. Huang, and L. Ke, "A dynamic and hierarchical access control for IoT in multi-authority cloud storage," Journal of Network and Computer Applications, vol. 160, pp. 102633, 2020.

[24] K. Guo, W. Pan, M. Lu, X. Zhou, and J. Ma, "An effective and economical architecture for semantic-based

DOI: 10.5281/zenodo.5226865
Received: March 19, 2021 Accepted: June 12, 2021

heterogeneous multimedia big data retrieval," Journal of Systems and Software, vol. 102, pp. 207 – 216, 2015.
[25] L. Xie, L. Zhu, and G. Chen, "Unsupervised multi-graph cross-modal hashing for large-scale multimedia retrieval," Multimedia Tools and Applications, vol. 75, no. 15, pp. 9185–9204, Aug 2016.
[26] K. Riad and L. Ke, "Secure storage and retrieval of IoT data based on private information retrieval," Wireless Communications and Mobile Computing, vol. 2018, pp. 1–10, 2018.
[27] K. Guo, Z. Liang, Y. Tang, and T. Chi, "Sor: An optimized semantic ontology retrieval algorithm for heterogeneous multimedia big data," Journal of Computational Science, vol. 28, pp. 455 – 465, 2018.
[28] Z. Gao, H. Zhang, G. Xu, Y. Xue, and A. Hauptmann, "Multi-view discriminative and structured dictionary learning with group sparsity for human action recognition," Signal Process., vol. 112, no. C, pp. 83–97, Jul. 2015.
[29] K. Li, G.-J. Qi, and K. A. Hua, "Learning label preserving binary codes for multimedia retrieval: A general approach," ACM Trans. Multimedia Comput. Commun. Appl., vol. 14, no. 1, pp. 2:1–2:23, Dec. 2017.
[30] J. Yan, H. Zhang, J. Sun, Q. Wang, P. Guo, L. Meng, W. Wan, and X. Dong, "Joint graph regularization-based modality-dependent cross-media retrieval," Multimedia Tools and Applications, vol. 77, no. 3, pp. 3009–3027, Feb 2018.
[31] C.-C. Yang, T.-Y. Chang, and M.-S. Hwang, "A (t,n) multi-secret sharing scheme," Applied Mathematics and Computation, vol. 151, no. 2, pp. 483– 490, 2004.
[32] OpenStack, "http://www.openstack.org/."
[33] H. Krawczyk, Advances in Cryptology, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, vol. 773, no. 0302-9743, ch. Secret Sharing Made Short, pp. 136–146.

DOI: 10.5281/zenodo.5226865
Received: March 19, 2021 Accepted: June 12, 2021