Enhancing Security of Cloud Computing with Replication

Prof. Ashru L. Korde¹, Prashant D. Shimp²
¹ME Co-Ordination, ²PG Student, KSIEET Hingoli.

Abstract— Cloud storage enables users to remotely store their data and enjoy the on-demand high quality cloud applications without the burden of local hardware and software management. Though the benefits are clear, such a service is also relinquishing users’ physical possession of their outsourced data, which inevitably poses new security risks toward the correctness of the data in cloud. In order to address this new problem and further achieve a secure and dependable cloud storage service, we propose in this paper a flexible distributed cloud storage integrity auditing mechanism, utilizing the homomorphic token and distributed erasure-coded data. The proposed design allows users to audit the cloud storage with very lightweight communication and computation cost. The auditing result not only ensures strong cloud storage correctness guarantee, but also simultaneously achieves fast data error localization, i.e., the identification of misbehaving server. Considering the cloud data are dynamic in nature, the proposed design further supports secure and efficient dynamic operations on outsourced data, including block modification, deletion, and append. Analysis shows the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

Index Terms—Epidemic Routing, Any path Routing Opportunistic networks, Probabilistic Routing.

I. INTRODUCTION

SEVERAL trends are opening up the era of cloud computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the Software as a Service (SaaS) computing architecture, are transforming data centers into pools of computing service on a huge scale. The increasing network bandwidth and reliable yet flexible network connections make it even possible that users can now subscribe high quality services from data and software that reside solely on remote data centers. Moving data into the cloud offers great convenience to users since they don’t have to care about the complexities of direct hardware management. The pioneer of cloud computing vendors, Amazon Simple Storage Service (S3), and Amazon Elastic Compute Cloud (EC2) [2] are both well-known examples. While these internet-based online services do provide huge amounts of storage space and customizable computing resources, this computing platform shift, however, is eliminating the responsibility of local machines for data maintenance at the same time.

As a result, users are at the mercy of their cloud service providers (CSP) for the availability and integrity of their data [3], [4]. On the one hand, although the cloud infrastructures are much more powerful and reliable than personal computing devices, broad range of both internal and external threats for data integrity still exist. Examples of outages and data loss incidents of noteworthy cloud storage services appear from time to time [5], [6], [7], [8], [9]. On the other hand, since users may not retain a local copy of outsourced data, there exist various incentives for CSP to behave unfaithfully toward the cloud users regarding the status of their outsourced data. For example, to increase the profit margin by reducing cost, it is possible for CSP to discard rarely accessed data without being detected in a timely fashion [10]. Similarly, CSP may even attempt to hide data loss incidents so as to maintain a reputation [11], [12], [13]. Therefore, although outsourcing data into the cloud is economically attractive for the cost and complexity of long-term large-scale data storage, its lacking of offering strong assurance of data integrity and availability may impede its wide adoption by both enterprise and individual cloud users. In order to achieve the assurances of cloud data integrity and availability and enforce the quality of cloud storage service, efficient methods that enable on-demand data correctness verification on behalf of cloud users have to be designed. However, the fact that users no longer have physical possession of data in the cloud prohibits the direct adoption of traditional cryptographic primitives for the purpose of data integrity protection. Hence, the verification of cloud storage correctness must be conducted without explicit knowledge of the whole data files [10], [11], [12], [13]. Meanwhile, cloud storage is not just a third party data warehouse. The data stored in the cloud may not only be accessed but also be frequently updated by the users [14], [15], [16], including insertion, deletion, modification, appending, etc. Thus, it is also imperative to support the integration of this dynamic feature into the cloud storage correctness assurance, which makes the system design even more challenging. Last but not the least, the deployment of cloud computing is powered by data centers running in a simultaneous, cooperated, and distributed manner [3]. It is more advantages for individual users to store their data redundantly across multiple physical servers so as to reduce the data integrity and availability threats.
Thus, distributed protocols for storage correctness assurance will be of most importance in achieving robust and secure cloud storage systems. However, such important area remains to be fully explored in the literature. Recently, the importance of ensuring the remote data integrity has been highlighted by the following research works under different system and security models [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22]. These techniques, while can be useful to ensure the storage correctness without having users possessing local data, are all focusing on single server scenario. They may be useful for quality-of-service testing [23], but does not guarantee the data availability in case of server failures. Although direct applying these techniques to distributed storage (multiple servers) could be straightforward, the resulted storage verification overhead would be linear to the number of servers. As an complementary approach, researchers have also proposed distributed protocols [23], [24], [25] for ensuring storage correctness across multiple servers or peers. However, while providing efficient cross server storage verification and data availability insurance, these schemes are all focusing on static or archival data. As a result, their capabilities of handling dynamic data remains unclear, which inevitably limits their full applicability in cloud storage scenarios. In this paper, we propose an effective and flexible distributed storage verification scheme with explicit dynamic data support to ensure the correctness and availability of users’ data in the cloud. We rely on erasure correcting code in the file distribution preparation to provide redundancies and guarantee the data dependability against Byzantine servers [26], where a storage server may fail in arbitrary ways. This construction drastically reduces the communication and storage overhead as compared to the traditional replication-based file distribution techniques. By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the storage correctness insurance as well as data error localization: whenever data corruption has been detected during the storage correctness verification, our scheme can almost guarantee the simultaneous localization of data errors, i.e., the identification of the misbehaving server(s). In order to strike a good balance between error resilience and data dynamics, we further explore the algebraic property of our token computation and erasure-coded data, and demonstrate how to efficiently support dynamic operation on data blocks, while maintaining the same level of storage correctness assurance.

In order to save the time, computation resources, and even the related online burden of users, we also provide the extension of the proposed main scheme to support third-party auditing, where users can safely delegate the integrity checking tasks to third-party auditors (TPA) and be worry-free to use the cloud storage services. Our work is among the first few ones in this field to consider distributed data storage security in cloud computing. Our contribution can be summarized as the following three aspects: 1) Compared to many of its predecessors, which only provide binary results about the storage status across the distributed servers, the proposed scheme achieves the integration of storage correctness insurance and data error localization, i.e., the identification of misbehaving server(s). 2) Unlike most prior works for ensuring remote data integrity, the new scheme further supports secure and efficient dynamic operations on data blocks, including: update, delete, and append. 3) The experiment results demonstrate the proposed scheme is highly efficient. Extensive security analysis shows our scheme is resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

II. DISADVANTAGES OF EXISTING SYSTEM

These techniques, while can be useful to ensure the storage correctness without having users possessing data, cannot address all the security threats in cloud data storage, since they are all focusing on single server scenario and most of them do not consider dynamic data operations.

As an complementary approach, researchers have also proposed distributed protocols for ensuring storage correctness across multiple servers or peers. Again, none of these distributed schemes is aware of dynamic data operations. As a result, their applicability in cloud data storage can be drastically limited.

III. PROPOSED SYSTEM

In this paper, we propose an effective and flexible distributed scheme with explicit dynamic data support to ensure the correctness of users’ data in the cloud. We rely on erasure correcting code in the file distribution preparation to provide redundancies and guarantee the data dependability. This construction drastically reduces the communication and storage overhead as compared to the traditional replication-based file distribution techniques.
By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the storage correctness insurance as well as data error localization: whenever data corruption has been detected during the storage correctness verification, our scheme can almost guarantee the simultaneous localization of data errors, i.e., the identification of the misbehaving server(s).

**Advantages Of Proposed System**

1. Compared to many of its predecessors, which only provide binary results about the storage state across the distributed servers, the challenge-response protocol in our work further provides the localization of data error.
2. Unlike most prior works for ensuring remote data integrity, the new scheme supports secure and efficient dynamic operations on data blocks, including: update, delete and append.
3. Extensive security and performance analysis shows that the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

**Working Of Proposed Architecture**

The process is carried out as follows:

**Step1:** Client sends data to cloud provider for storing.
**Step2:** Cloud provider receives data and performs encryption.
**Step3:** Full copy of encrypted data stores on data warehouse.
**Step4:** After backup, performing replication and divide the data in parts according to the availability of data marts (in our system use three data marts S1, S2, S3)
**Step5:** Storing the different part of information on different data mart.
**Step6:** Repeat Steps as per storing request

**Recovery of Lost Information**

1) **Software failure**

In case of software failure data mart loss the information of particular client due to different causes like network down, file outage etc. Data mart takes the copy from backup warehouse. It increases the availability of information. Data mart S1 lost the information P1, then it can take information S1 from backup warehouse and reconfigure it. If data mart S2 and S3 lost the information then they can also able to recover the information from backup warehouse.

2) **Hardware failure**

The data mart is crashes or down also impact on the availability of information. The purposed system also removes that drawback. If any data mart is crashes or down then client’s request also able to extract the data from backup warehouse. In Purposed scenario data mart S1 is fail and not responding the user request. In this case the part of information P1 is lost.
The purposed system allow user to extract the information from backup ware house. The availability of data mart also affect on security of information. In case of large no of data marts the data divide in more parts and store different parts in different data marts. Each data mart have very small part of information. If any data mart is hacked by attacker then it can take only small part of information.

**Hybrid Pair Scheme:**

When we run the application, a login form turn up, allowing the user to enter the username. The form appeared consists of three buttons-register, login, close.

If the user is already a registered one, then clicking on the “login”button would advance him to the second phase of the application. If the user is not a registered member, then on doing the above action would generate a message box conveying “username doesn’t exist”. Thus, in order to make use of the application, the person must get registered by the admin.

Consequently, on clicking the “register”button on the login form would display a window allowing the user to enter his mobile number. On submitting the mobile number, a password is directed to his mobile by the admin.

Then, the user has to write down the password in the interface shown subsequently, and has to click on the “register” button below that interface. On doing so, registration form appears.

Thereafter, the user enters textual password whose minimum length is 8 and it contains even number of characters. If the user violates this protocol, then a message box expressing the fault with the textual password is displayed. This password is to be remembered as pair-based password, also known as secret pass. Besides this, the user has to rank the colors portrayed as color grid of 8 colors in the registration form. The rank (from 1 to 8) associated with each color has to be remembered as the hybrid textual password.

Along with these, graphical password (draw-a-secret) using the 3X3 grid is sketched. Moreover, basic details like first name, last name, mobile number and email-id are submitted by the user.

Clicking the “new” button in the registration form would automatically generate the user-id based on the existing users. On ticking the “save”button, all the information inserted by the user is stored on to the database. Thereby, again the login form is displayed, where the user now clicks on “login”button advancing him to the second phase of the application.

In the course of registration, the user submits the secret pass. The minimum length of the secret pass is 8 and it should contain even number of characters.

During the primary level authentication, when the user chooses the pair-based authentication scheme, an interface consisting of 6X6 grid is displayed. The grid contains both alphabets and numbers which are placed at random and the interface changes every time.

The mechanism involved in the pair-based authentication scheme is as follows: Firstly, the user has to consider the secret pass in terms of pairs. The first letter in the pair is used to select the row and the second letter is used to select the column in the 6X6 grid. The intersection letter of the selected row and column generates the character which is a part of the session password. In this way, the logic is reiterated for all other pairs in the secret pass [4]. Thereafter, the password inputted by the user i.e. the session password is now verified by the server to authenticate the user.

**Splitting Data:**

In this paper, we propose a new symmetric solution for two-server PAKE. In all existing two-server PAKE protocols, two servers are provided random password shares pw1 and pw2 subject to pw1 + pw2 = pw. The system we are going to developed is capable to remove the disadvantages of existing system and produced the more efficient system. Our protocol can be applied in distributed systems where multiple servers exist. For example.

Microsoft active directory domain service (ADDS) is the foundation for distributed networks built on Windows server operating systems that use domain controllers. AD DS provides structured and hierarchical data storage for objects in a network such as users, computers, printers, and services.ADS also provides support for locating and working with these objects. For a large enterprise running, its own domain, there must be two AD DS domain controllers, for fault-tolerance purpose. To authenticate a user on a network, the user usually needs to provide his/her identification and password to one AD DS domain controller. Based on our two-server PAKE protocol, we can split the users password into two parts and store them, respectively, on the two AD DS domain controllers, which can then cooperate to authenticate the user. Even if one domain controller is compromised, the system can still work. In this way, we can achieve more secure AD DS.

The remainder of this project is organized as follows: two cryptographic building blocks of our protocol, Diffie-Hellman key exchange protocol and El-Gamal encryption scheme and two-server PAKE protocol.
Our Protocol Works in Four Phases:

A. Initialization

To secure hash function

\[ H : 0; 1^2q \]

The two servers S1 and S2 jointly decide a cyclic group G of large prime order q with a generator g1 which maps a message of random length into an l-bit integer, where \( l = \log_2 q \). After that, S1 chooses an integer s1 from \( Z^*_q \) randomly, and S2 chooses an integer s2 from \( Z^*_q \) randomly, and S1 and S2 exchange g1s1 and g1s2.

Next, S1 and S2 together publish public system parameters G,q, g1,g2,H where \( g2 = g1s1s2 \).

B. Registration

For authentication, each client C is need to register both server S1 and S2 through unlike secure channels. Firstly, the client C generates encryption and decryption key pairs (xi; yi) using the public parameters published by the two servers where \( y_i = g^x_i \) for the server Si (i=1).

After that, client C selects a password pwC and encrypts that password by using the encryption key yi, i.e., (g2pwC; y1) = (Ai,Bi) = (g1ai, g2pwcyi) (i=1,2) where ai is chosen randomly from \( Z^*_q \) according to El-Gamal encryption. After that, the client C chooses b1 randomly from \( Z^*_q \) and lets \( b2 = H(pwC) b1 \), where stands for two l-bit blocks exclusive OR. Finally, client C sends the password authentication information to S1 through a secure channel, i.e. Authc(1) = x1; a1; b1; (g2pwC; y2) and the password authentication information to S2 through another secure channel i.e. Authc(2) = x2; a2; b2; (g2pwC; y1): Next, client C remembers the only password pwC.

C. Authentication and Key Exchange

Now we consider that the two servers S1 and S2 having the authentication information of a client C, to authenticate the client C there are five steps for the S1 and S2 and establish private session keys with the client C in terms of parallel computation.

D. User Authentication Module

Here system having all details of user likes username and password. Now here system will call both data servers and take the encrypted data paces, then apply the same algorithm but here is reverse process known as decryption.

The decrypted data will be match with user password we already store in virtual data table then user will be authenticated.

Input: User login details.
Output: User login success or login failed.

IV. CONCLUSION

In this paper, we investigate the problem of data security in Centralized Server data storage, which is essentially a distributed storage system. To achieve the assurances of Centralized Server data integrity and availability and enforce the quality of dependable Centralized Storage service for users, we propose an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. We rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. By utilizing the homomorphic token with distributed verification of erasure coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server(s). Considering the time, computation resources, and even the related online burden of users, we also provide the extension of the proposed main scheme to support third-party auditing, where users can safely delegate the integrity checking tasks to third party auditors and be worry-free to use the Centralized Storage services. Through detailed security and extensive experiment results, we show that our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks.

REFERENCES

International Journal of Emerging Technology and Advanced Engineering


