International Journal on Applications in Information and Communication Engineering
Volume 1: Issue 9: September 2015, pp 56-60. www.aetsjournal.com ISSN (Online) : 2394-6237

Unique Identity Based Provable Data Control in Multi Cloud Environment

J.Emily Helda , D.Jebasudha

Abstract— Remote data integrity checking is of crucial importance in cloud storage. It can make the clients verify whether their outsourced data is kept intact without downloading the whole data. In some application scenarios, the clients have to store their data on multi-cloud servers. At the same time, the integrity checking protocol must be efficient in order to save the verifier's cost. From the two points, we propose a novel remote data integrity checking model: ID-DPDP (Identity-Based Distributed Provable Data Possession) in multi-cloud storage. The formal system model and security model are given. Based on the bilinear pairings, a concrete ID-DPDP protocol is designed. The proposed ID-DPDP protocol is provably secure under the hardness assumption of the standard CDH (computational Diffie-Hellman) problem. In addition to the structural advantage of Elimination of certificate management, our ID-DPDP protocol is also efficient and flexible. Based on the client's authorization, the proposed ID-DPDP protocol can realize private verification, delegated verification and public verification.

Index Terms— Cloud storage, Remote data integrity, Multi-cloud servers, Identity-based distributed provable data possession (ID-DPDP).

I. INTRODUCTION

Advances in networking and computing technologies have prompted many organizations to outsource their storage needs on demand. This new economic and computing paradigm is commonly referred to as cloud storage. It brings appealing benefits including relief of the burden for storage management, universal data access with independent geographical locations, and avoidance of capital expenditure on hardware, software, and personnel maintenances, etc. However, there are barriers that hinder migration to the cloud.

One of the main barriers is that, due to lack of physical control over the outsourced data, a cloud user may worry about whether her data are kept as expected. If the cloud user is a company, apart from the risk of remote malicious attacks on the cloud, the traditional concerns posed by malicious company insiders are now supplemented by the even more hazardous threat of malicious outsiders who are given the power of insiders. A recent EU bill forces companies migrating to the cloud to be liable for any data corruption or privacy breach into which their cloud service provider (CSP) may incur, even when they do not retain control over their data. Convincing cloud users that their data are intact is especially vital when users are companies. Remote data possession checking (RDPC) is a primitive designed to address this issue.

A. Remote Data Possession Checking

RDPC allows a client that has stored data at a public cloud server (PCS) to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof. The challenge/response protocol transmits a small, constant amount of data, which minimizes network communication. In order to achieve secure RDPC implementations, Ateniese et al. proposed a provable data possession (PDP) paradigm and designed two provably-secure PDP3 schemes based on the difficulty of large integer factoring. They refined the original paradigm and proposed a dynamic PDP scheme in but their proposal does not support the insert operation. In order to solve this problem, Erway et al. proposed a full-dynamic PDP scheme by employing an authenticated flip table. Following Ateniese et al.'s pioneering work, researchers devoted great efforts to RDPC with extended models and new protocols. One of the variations is the proof of retrievability (POR), in which a data storage server cannot only prove to a verifier that he is actually storing all of a client's data, but also it can prove that the users can retrieve them at any time. This is stronger than the regular PDP notion. Shacham presented the first POR schemes with provable security. The state of the art can be found in but few POR protocols are more efficient than their PDP counterparts. The challenge is to build POR systems that are both efficient and provably secure. Note that one of benefits of cloud storage is to enable universal data access with independent geographical locations. This implies that the end devices may be mobile and limited in computation and storage. Regular RDPC protocols are more suitable for cloud users equipped with mobile end devices. Our ID-RDPC architecture and protocol are based on the PDP model.

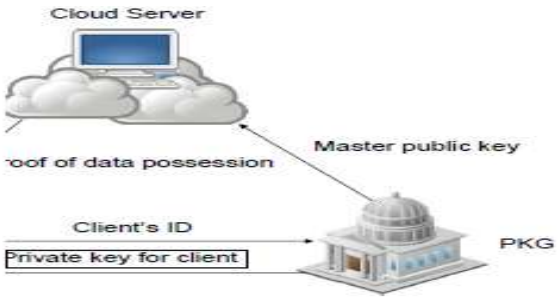
B. Motivation and Contribution

This approach focuses on RDPC in company-oriented cloud storage. Consider a scenario in which a company purchases the cloud storage service. Only the staff members of the company are allowed to upload data to the PCS and may check the integrity of their data with mobile devices. PCS has to be convinced that the data (and their tags) to be uploaded come from the staff of the company, although this step is

J.Emily Helda, PG Scholar, Department of Computer Science and Engineering, Mount Zion College of Engineering & Technology Pudukottai
D.Jebasudha , Assistant Professor, Department of Computer Science and Engineering, Mount Zion College of Engineering & Technology Pudukottai

usually omitted in the existing models. The metadata or the tags are indeed signatures of the original data. Note that the existing RDPC protocols are designed in the PKI setting. PCS needs to validate the tags and the appended public key certificate of the users. The validation of the legit uploads incurs considerable overheads since the staff may 4 frequently upload data to PCS. This burden can only be partially mitigated by letting PCS cache the verified certificates. Indeed, caching cannot be used for certificates revoked before their expiration, for employees who leave the company, for newly recruited employees, etc. In addition to the heavy certificate verification, the system suffers from complicated certificate management: certificates generation, delivery, revocation, renewal, etc. In order to solve the above problem, we investigate a new RDPC model incorporating identity based cryptography, i.e., the ID-RDPC model. Our contribution is twofold:

- First, we formalize the ID-RDPC model. In this model, a trusted private key generator (PKG) generates the system public key and the master secret key. The PKG also generates private keys for the clients, i.e., the staff members of the company, by taking as input the staff members' identities and the PKG's master secret key. With a private key, the client can generate the tags of the data to be uploaded. Upon receiving a request of a data possession proof, the PCS can generate the proof without verifying any certificate but simply checking that the corresponding system public key is from a company allowed to use the service. Finally, the client can verify whether the PCS-generated proof is valid.
- Second, we realize the first ID-RDPC protocol. The main challenge to design the ID-RDPC protocol is that it requires the client to generate aggregately ID-based signatures like tags without applying the hash-and-sign paradigm to the original data. We address this with a variation of the well-known Schnorr signature. The instantiated ID-RDPC protocol is shown to be secure by assuming the hardness of the Computational Diffie-Hellman problem. In addition to the structural advantage of eliminating certificate management and verification, our ID-RDPC protocol is more efficient than the existing RDPC protocols in the PKI setting in terms of computation and communication.



. Fig 1: The System Model of ID-RPDC

C. Modeling of ID-RPDC

The ID-RDPC system model and its security definition are given in this section. An ID-RDPC protocol comprises three different entities, as illustrated in Fig.1.2. They can be identified as follows:

A) PKG (Private Key Generator): Entity, trusted by the clients and the PCSs, that generates the public parameters Params, the master public key mpk, the master secret key msk and the private key of the Client defined below.

B) Client: Entity which has massive data to be stored on the public cloud for maintenance and computation. Clients can be either individual consumers or group consumers, e.g., the departments of the company in the motivated scenario.

C) Cloud Server: Entity, managed by the cloud service provider that has significant storage space and computational resources to maintain the clients' data.

In the cloud paradigm, by putting the large data files on the remote cloud servers, the clients can be relieved of the burden of storage and computation. As the clients no longer possess their data locally, it is of critical importance for them to ensure that their data are being correctly stored and maintained. That is, clients should be equipped with certain security means so that they can periodically verify the correctness of the remote data even without the existence of local copies.

II. Related Wirk

We introduce a model for provable data possession (PDP) that allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof [1]. The challenge/response protocol transmits a small, constant amount of data, which minimizes network communication. Thus, the PDP model for remote data checking supports large data sets in widely-distributed storage systems.

We present two provably-secure PDP schemes that are more efficient than previous solutions, even when compared with schemes that achieve weaker guarantees. In particular, the overhead at the server is low (or even constant), as opposed to linear in the size of the data. Experiments using our implementation verify the practicality of PDP and reveal that the performance of PDP is bounded by disk I/O and not by cryptographic computation.

The scalable and efficient provable data possession (SEPDP) library allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it or storing a copy himself. The scheme using symmetric key constructs, making it a computationally efficient [2].

As storage-outsourcing services and resource-sharing networks have become popular, the problem of efficiently proving the integrity of data stored at untrusted servers has received increased attention. In the provable data possession (PDP) model, the client preprocesses the data and then sends it to an untrusted server for storage, while keeping a small amount of meta-data [3]. The client later asks the server to

prove that the stored data has not been tampered with or deleted (without downloading the actual data). However, the original PDP scheme applies only to static (or append-only) files. We present a definitional framework and efficient constructions for dynamic provable data possession (DPDP), which extends the PDP model to support provable updates to stored data. We use a new version of authenticated dictionaries based on rank information. The price of dynamic updates is a performance change from O (1) to O (log n) (or O (n ǫ log n)), for a file consisting of n blocks, while maintaining the same (or better, respectively) probability of misbehavior detection. Our experiments show that this slowdown is very low in practice (e.g., 415KB proof size and 30ms computational overhead for a 1GB file). We also show how to apply our DPDP scheme to outsourced file systems and version control systems (e.g., CVS).

Checking data possession in networked information systems such as those related to critical infrastructures (power facilities, airports, data vaults, defense systems, etc.) is a matter of crucial importance [4]. Remote data possession checking protocols permit to check that a remote server can access an uncorrupted file in such a way that the verifier does not need to know beforehand the entire file that is being verified. Unfortunately, current protocols only allow a limited number of successive verifications or are impractical from the computational point of view. In this paper, we present a new remote data possession checking protocol such that: 1) it allows an unlimited number of file integrity verifications; 2) its maximum running time can be chosen at set-up time and traded off against storage at the verifier.

Provable data possession (PDP) is a technique for ensuring the integrity of data in outsourcing storage service. In this project, we address the construction of efficient PDP schemes on hybrid clouds to support scalability of service and data migration, in which we consider the existence of multiple cloud service providers (CSP) to cooperatively store and maintain the clients' data. The proposed PDP schemes include an interactive PDP (IPDP) and cooperative PDP (CPDP) schemes adopting zero-knowledge property and three-layered index hierarchy, respectively. In particular, we present an efficient method for selecting the optimal number of sectors in each block to minimize the computation costs of clients and storage service providers [5]. Our experiments show that the verification requires a small, constant amount of overhead, which minimizes communication complexity.

III. PROPOSED SCHEME

First, we analyze the performance of our proposed ID-DPDP protocol from the computation and communication overhead. We compare our ID-DPDP protocol with the other up-to date PDP protocols. Second, we analyze our proposed ID-DPDP protocol's properties of flexibility and verification. Third, we give the prototypal implementation of the proposed ID-DPDP protocol.

The signature relates the client's identity with his private key. Distributed computing is used to store the client's data on multi-cloud servers. At the same time, distributed computing is also used to combine the multi-cloud servers' responses to respond the verifier's challenge.

Advantages

- In our proposed system each client has a private correspond to his identity (i.e.) name, id or any…
- The public verifier allow the user to correspond to his identity (i.e.) private Key

A. Heterogeneity and Tight Coupling

Clouds implement proprietary interfaces for service access, configuration, and management as well as for interaction with other cloud components. Each service layer of a cloud tightly integrates with lower service layers or is highly dependent on the value-added proprietary solutions that the cloud offers. This heterogeneity and tight coupling prohibit interoperation between services from different clouds. The current business model requires preestablished agreements between CSPs before collaboration can occur. These agreements are necessary for clouds to establish their willingness to collaborate and establish trust with one another. The lack of such agreements prohibits multicloud collaborative efforts due to incompatible intentions, business rules, and policies. Moreover, collaborations resulting from preestablished agreements typically exhibit tight integration between the participants and cannot be extended to provide universal and dynamic collaboration.

B. Key Generation Algorithm (KeyGen)

Key Generation algorithm that is run by the user to setup the scheme. Generating keys (Based on Hint Words) and mail it to users for decrypting the encrypted data. Key generation is the process of generating keys for cryptography. A key is used to encrypt and decrypt whatever data is being encrypted or decrypted. Modern cryptographic systems include symmetric-key algorithms (such as DES and AES) and public-key algorithms (such as RSA). Symmetric-key algorithms use a single shared key; keeping data secret requires keeping this key secret. Public-key algorithms use a public key and a private key. The public key is made available to anyone (often by means of a digital certificate). A sender encrypts data with the public key; only the holder of the private key can decrypt this data. Since public-key algorithms tend to be much slower than symmetric-key algorithms, modern systems such as TLS and SSH use a combination of the two: one party receives the other's public key, and encrypts a small piece of data (either a symmetric key or some data used to generate it). The remainder of the conversation uses a (typically faster) symmetric-key algorithm for encryption. Computer cryptography uses integers for keys. In some cases keys are randomly generated using a random number generator (RNG) or pseudorandom number generator (PRNG). A PRNG is a computer algorithm that produces data that appears random under analysis. PRNGs that use system entropy to seed data generally produce better results, since this makes the initial conditions of the PRNG much more difficult for an attacker to guess. In other situations, the key is created using a passphrase

and a key generation algorithm, usually involving a cryptographic hash function such as SHA-1. The simplest method to read encrypted data is a brute force attack—simply attempting every number, up to the maximum length of the key. Therefore, it is important to use a sufficiently long key length; longer keys take exponentially longer to attack, rendering a brute force attack impractical. Currently, key lengths of 128 bits (for symmetric key algorithms) and 1024 bits (for public-key algorithms) are common.

C. Signature Generation Algorithm (SigGen)

Used by the user to generate verification metadata, which may consist of unique signatures or other information used for verifying the user? Signature Generation Algorithm is used by the user to generate verification metadata, which may consist of machine access code. This algorithm checks the signatures, or other related information that will be used for auditing. It generates the signature for the user and set the identity to each and every individual in the cloud architecture.

IV. SYSTEM DESIGN

A. System Architecture

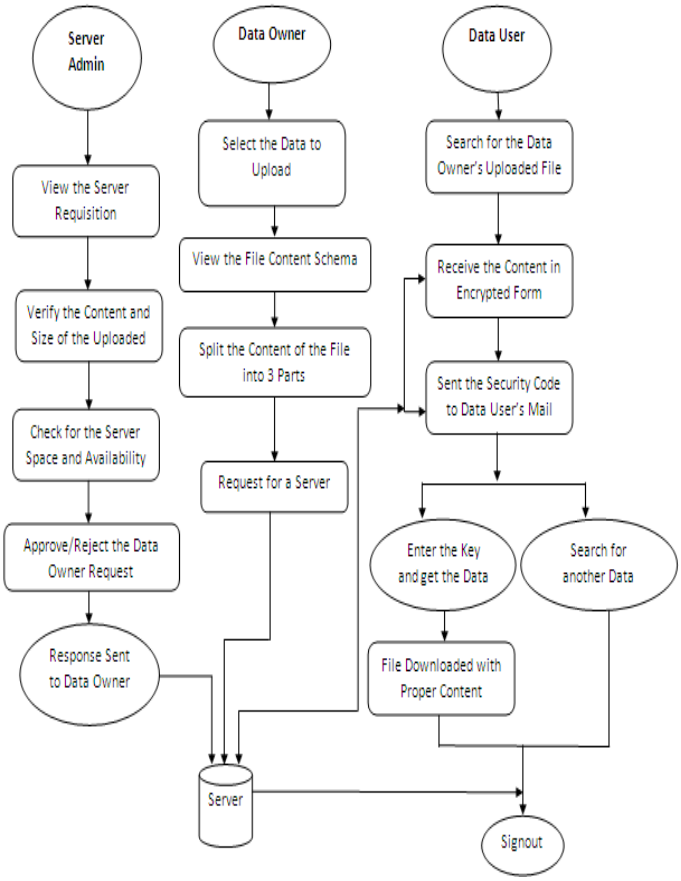


Fig 2: System Architecture

The major part of the project development sector considers and fully survey all the required needs for developing the project. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations. Generally algorithms shows a result for exploring a single thing that is either be a performance, or speed, or accuracy, and so on. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

Clouds implement proprietary interfaces for service access, configuration, and management as well as for interaction with other cloud components. Each service layer of a cloud tightly integrates with lower service layers or is highly dependent on the value-added proprietary solutions that the cloud offers. This heterogeneity and tight coupling prohibit interoperation between services from different clouds. The current business model requires pre-established agreements between CSPs before collaboration can occur. These agreements are necessary for clouds to establish their willingness to collaborate and establish trust with one another. The lack of such agreements prohibits multi-cloud collaborative efforts due to incompatible intentions, business rules, and policies. Moreover, collaborations resulting from pre-established agreements typically exhibit tight integration between the participants and cannot be extended to provide universal and dynamic collaboration.

V. METHODOLOGY

Following are the most frequently used project management Methodologies in the project management practice:

A. Secure Key Processing
B. Verification Generator
C. Server Data Processing
D. Data Assurance to Admin Process
E. Admin Auditing Model

A. Secure Key Processing

The Secure Key Processing module adds the facility to the site to create the random set of keys to verify the user identity as well as the data identity by means of a Key Generation algorithm that is run by the user to setup the scheme. Generating keys (Based on Hint Words) and mail it to users for decrypting the encrypted data. Key generation is the process of generating keys for cryptography. A key is used to encrypt and decrypt whatever data is being encrypted or decrypted. Modern cryptographic systems include symmetric-key algorithms (such as DES and AES) and public-key algorithms (such as RSA). Symmetric-key algorithms use a single shared key; keeping data secret requires keeping this key secret. Public-key algorithms use a public key and a private key. The public key is made available to anyone (often by means of a digital certificate). A sender encrypts data with the public key; only the holder of the private key can decrypt this data. Since public-key algorithms tend to be much slower than symmetric-key algorithms, modern systems such as TLS

and SSH use a combination of the two: one party receives the other's public key, and encrypts a small piece of data (either a symmetric key or some data used to generate it). The remainder of the conversation uses a (typically faster) symmetric-key algorithm for encryption. Computer cryptography uses integers for keys. In some cases keys are randomly generated using a random number generator (RNG) or pseudorandom number generator (PRNG). A PRNG is a computer algorithm that produces data that appears random under analysis. PRNGs that use system entropy to seed data generally produce better results, since this makes the initial conditions of the PRNG much more difficult for an attacker to guess.

A. Verification Generator

The verification generator module allows the system to generate the verification code / signature for the users to securely handle the data in a remote medium. It is used by the user to generate verification metadata, which may consist of unique signatures or other information used for verifying the user? Signature Generation Algorithm is used by the user to generate verification metadata, which may consist of machine access code. This algorithm checks the signatures, or other related information that will be used for auditing. It generates the signature for the user and set the identity to each and every individual in the cloud architecture.

B. Server Data Processing

The server data processing module fully describes about the cloud implementation process. Clouds implement proprietary interfaces for service access, configuration, and management as well as for interaction with other cloud components. Each service layer of a cloud tightly integrates with lower service layers or is highly dependent on the value-added proprietary solutions that the cloud offers. This heterogeneity and tight coupling prohibit interoperation between services from different clouds. The current business model requires pre-established agreements between CSPs before collaboration can occur. These agreements are necessary for clouds to establish their willingness to collaborate and establish trust with one another. The lack of such agreements prohibits multi-cloud collaborative efforts due to incompatible intentions, business rules, and policies. Moreover, collaborations resulting from pre-established agreements typically exhibit tight integration between the participants and cannot be extended to provide universal and dynamic collaboration..

C. Data Assurance to Admin Process

The data assurance module provides the facility to the administrator to check the resource owner want to upload into the server is valid or not. If the owner requesting for the proper resource to upload it will be verified by the administrator and get the permission properly and get splitted into three parts and stored into various servers for providing the security means, but if the requesting resource upload permission is for the wrong resource then it will be blocked by the administrator immediately and the owner cannot be upload the resource further

D. Admin Auditing Model

The administrator auditing panel allows the administrator to audit the resource which is uploaded by the resource owners, in which the process is also known as public verifier process. The public verifier is able to correctly check the integrity of shared data. The public verifier can audit the integrity of shared data from multi-cloud with whole data and accept the file. The public auditor checks all files integrity and accept the files to cloud server for further process like searching and maintenance.

II. Conclusion

Favorable solutions to ensure data privacy must employ flexible data perturbation methods that provide control over the tradeoff between the privacy guarantee and the utility of the query results. Prevent dynamic data integrity among applications hosted by different cloud systems. Proxy services are implemented to maintain the authentication and initially provide support for simple use cases, later progressing to more complex use cases.

References

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song. Provable Data Possession at Untrusted Stores. CCS'07, pp. 598-609, 2007.
[2] G. Ateniese, R. DiPietro, L. V. Mancini, G. Tsudik. Scalable and Efficient Provable Data Possession. Secure Comm. 2008, article 9, 2008.
[3] C. C. Erway, A. Kupcu, C. Papamanthou, R. Tamassia. Dynamic Provable Data Possession. CCS'09, 213-222, 2009.
[4] F. Seb´e, J. Domingo-Ferrer, A. Martınez-Ballest´e, Y. Deswarte, J. Quisquater. Efficient Remote Data Integrity checking in Critical Information Infrastructures. IEEE Transactions on Knowledge and Data Engineering, 20(8):1034-1038, 2008.
[5] Y. Zhu, H. Wang, Z. Hu, G. J. Ahn, H. Hu, S. S. Yau. Efficient Provable Data Possession for Hybrid Clouds. CCS'10, 756-758, 2010.
[6] Y. Zhu, H. Hu, G.J. Ahn, M. Yu. Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage. IEEE Transactions on Parallel and Distributed Systems, 23(12):2231-224, 2012.
[7] R. Curtmola, O. Khan, R. Burns, G. Ateniese. MR-PDP: Multiple-Replica Provable Data Possession. ICDCS'08, 411-420, 2008.