A Novel Technique for Auditing Through Key Exposure on Cloud Storage

M.Prasoona[1], I.Kavitha Jackleen[2]
[1]*M.Tech (CSE)., Dept of CSE, Global College Of Engineering and Technology, Kadapa, Andhra Pradesh,*
[2]*Assistant Professor, Dept of CSE, Global College Of Engineering and Technology, Kadapa, Andhra Pradesh*

Abstract- **Key-exposure resistance has always been an important issue for in-depth cyber defence in many security applications. Recently, how to deal with the key exposure problem in the settings of cloud storage auditing has been proposed and studied. To address the challenge, existing solutions all require the client to update his secret keys in every time period, which may inevitably bring in new local burdens to the client, especially those with limited computation resources, such as mobile phones. In this paper, we focus on how to make the key updates as transparent as possible for the client and propose a new paradigm called cloud storage auditing with verifiable outsourcing of key updates. In this paradigm, key updates can be safely outsourced to some authorized party, and thus the key-update burden on the client will be kept minimal. In particular, we leverage the third party auditor (TPA) in many existing public auditing designs, let it play the role of authorized party in our case, and make it in charge of both the storage auditing and the secure key updates for keyexposure resistance. In our design, TPA only needs to hold an encrypted version of the client's secret key while doing all these burdensome tasks on behalf of the client. The client only needs to download the encrypted secret key from the TPA when uploading new files to cloud. Besides, our design also equips the client with capability to further verify the validity of the encrypted secret keys provided by the TPA. All these salient features are carefully designed to make the whole auditing procedure with key exposure resistance as transparent as possible for the client. We formalize the definition and the security model of this paradigm. The security proof and the performance simulation show that our detailed design instantiations are secure and efficient.**

Index Terms- **Cloud storage, outsourcing computing, cloud storage auditing, key update, verifiability.**

I. INTRODUCTION

Cloud computing, as a new technology paradigm with promising further, is becoming more and more popular nowadays. It can provide users with seemingly unlimited com-puting resource. Enterprises and people can outsource time-consuming computation workloads to cloud without spending the extra capital on deploying and maintaining hardware and software. In recent years, outsourcing computation has attracted much attention and been researched widely. It has been considered in many applications including scientific computations [1], linear algebraic computations [2], linear programming computations [3] and modular exponentiation computations [4], etc. Besides, cloud computing can also provide users with seemingly unlimited storage resource. Cloud storage is universally viewed as one of the most important services of cloud computing. Although cloud stor-age provides great benefit to users, it brings new security challenging problems. One important security problem is how to efficiently check the integrity of the data stored in cloud. In recent years, many auditing protocols for cloud storage have been proposed to deal with this problem. These pro-tocols focus on different aspects of cloud storage auditing such as the high efficiency [5]–[17], the privacy protec-tion of data [18], the privacy protection of identities [19], dynamic data operations [13], [15], [16], [20], the data sharing [21], [22], etc. The key exposure problem, as another important problem in cloud storage auditing, has been considered [23] recently. The problem itself is nontrivial by nature. Once the client's secret key for storage auditing is exposed to cloud, the cloud is able to easily hide the data loss incidents for maintaining its reputation, even discard the client's data rarely accessed for saving the storage space. Yu et al. [23] constructed a cloud storage auditing protocol with key-exposure resilience by updating the user's secret keys periodically. In this way, the damage of key exposure in cloud storage auditing can be reduced. But it also brings in new local burdens for the client because the client has to execute the key update algorithm in each time period to make his secret key move forward. For some clients with limited computation resources, they might not like doing such extra computations by themselves in each time period. It would be obviously more attractive to make key updates as transparent as possible for the client, especially in frequent key update scenarios. In this paper, we consider achieving this goal by outsourcing key updates. However, it needs to satisfy several new requirements to achieve this goal. Firstly, the real client's secret keys for cloud storage auditing should not be known by the authorized party who performs outsourcing computation for key updates Otherwise, it will bring the new security threat. So the authorized party should only hold an encrypted version of the user's secret key for cloud storage auditing. Secondly, because the authorized party performing outsourcing computation only knows the encrypted secret keys, key updates should be completed under the encrypted state. In other words, this authorized party should be able to update secret keys for cloud storage auditing from the encrypted version he holds. Thirdly, it should be very efficient for the client to recover the real secret key from the encrypted version that is retrieved from the authorized party. Lastly, the client should be able to verify the validity of the encrypted secret key after the client retrieves it from the authorized party. The goal of this paper is to design a cloud storage auditing protocol that can satisfy above requirements to achieve the outsourcing of key updates. The main contributions are as follows: (1) We propose a new paradigm called cloud storage auditing with verifiable outsourcing of key updates. In this new paradigm, key-update operations are not performed by the client, but by an authorized party. The authorized party holds an encrypted secret key of the client for cloud storage auditing and updates it under the encrypted state in each time period. The client downloads the encrypted secret key from the authorized party and decrypts it only when he would like to upload new files to cloud. In addition, the client can verify the validity of the encrypted secret key.

II RELATED WORK

Outsourcing Computation: How to effectively outsource time-consuming computations has become a hot topic in the research of the theoretical computer science in the recent two decades. Outsourcing computation has been considered in many application domains. Chaum and Pedersen [24] firstly proposed the notion of wallet databases with observers, in which a hardware was used to help the client perform some expensive computations. The method for secure out-sourcing of some scientific computations was proposed by Atallah et al. [1]. Chevallier-Mames et al. [25] designed the first effective algorithm for secure delegation of ellipticcurve pairings based on an untrusted server. The first out-sourcing algorithm for modular exponentiations was proposed by Hohenberger and Lysyanskaya [26], which was based on the methods of precomputation and server-aided compu-tation. Atallah and Li [27] proposed a secure outsourcing algorithm to complete sequence comparisons. Chen et al. [4] proposed new algorithms for secure outsourcing of modular exponentiations. Benjamin and Atallah [2] researched on how to securely outsource the computation for linear algebra. Atallah and Frikken [28] gave further improvement based on the weak secret hiding assumption. Wang et al. [3] presented an efficient method for secure outsourcing of linear programming computation. Chen et al. [29] proposed an outsourc-ing algorithm for attribute-based signatures computations. Zhang et al. [30] proposed an efficient method for outsourcing a class of homomorphic functions. Cloud Storage Auditing: How to check the integrity of the data stored in cloud is a hot topic in cloud security. The notion of "provable data possession" (PDP) was firstly proposed by Ateniese et al. [5] to ensure data possession at untrusted servers. The notion of "proof of retrievability" (PoR) was proposed by Juels et al. [6] to ensure both possession and retriev-ability of data at untrusted servers. Wang et al. [18] proposed a public privacy-preserving auditing protocol. They used the ran-dom masking technique to make the protocol achieve privacy-preserving property. Proxy provable data possession protocol was proposed in [17]. The auditing protocols supporting dynamic data operations were also proposed in [13] and [20]. Yang and Jia [16] proposed an auditing protocol supporting both the dynamic property and the privacy preserving property. The privacy preserving of the user's identity for shared data auditing was considered in [19]. The problem of user revocation in shared data auditing was considered in [21]. Yuan and Yu [22] proposed a public auditing protocol for data sharing with multiuser modification. Sookhak et al. [31] proposed a public cloud auditing protocol for securing big data storage based on algebraic signature. Guan et al. [32] proposed the first cloud storage auditing protocol based on indistinguishability obfuscation, which is especially useful for low-power cloud users. Yang et al. [33] proposed a public auditing protocol for shared cloud data supporting both iden-tity privacy and identity traceability. All above auditing protocols are all built on the assumption that the secret key of the client is absolutely secure and would not be exposed. In [23], the authors firstly considered the key exposure problem in cloud storage auditing and proposed a cloud storage auditing protocol with key-exposure resilience. In that protocol, the secret keys for cloud storage auditing are updated periodically. As a result, any dishonest behaviors, such as deleting or modifying the client's data previously stored in cloud, can all be detected, even if the cloud gets the client's current secret key for cloud storage auditing. However, the client needs to update his secret key in each time period. It will add obvious computation burden to the client, especially when key updates are very frequent.

III. PROPOSED ARCHITECTURE

In this paper, we concentrate on the best way to make the key overhauls as straightforward as could be expected under the circumstances for the customer and propose another worldview called distributed storage reviewing with certain outsourcing of key redesigns. In this worldview key overhauls can be securely outsourced to some approved gathering and along these lines the key-upgrade trouble on the customer will be kept insignificant. In particular, we influence the outsider inspector (TPA) in numerous current open examining outline, let it assume the part of approved gathering for our situation and make it accountable for both the capacity reviewing and secure key upgrades for key-presentation resistance. they are not generated the particular key of any file means one file are only on e key are generated In our outline, TPA just needs to hold a scrambled variant of the customer's mystery key, while doing all these difficult assignments for the benefit of the customer. The customer just needs to download the scrambled mystery key from the TPA while transferring new documents to cloud. Moreover, our plan additionally outfits the customer with capacity to facilitate confirm the legitimacy of the scrambled mystery keys gave by TPA. We formalize the definition and the security model of this worldview. The security confirmation and the execution reenactment demonstrate that our point by point plan instantiations are secure and productive.

The TPA does not know the real secret key of the client for cloud storage auditing, but only holds an encrypted version. In the detailed protocol we use the blinding technique with homomorphism property to form the encryption algorithm to encrypt the secret key held by the TPA.it makes our protocol secure and the decryption operation efficient.

2. Meanwhile, The TPA can complete key updates under the encrypted state. The Client can validity of the encrypted secret key when he retrieve it from the TPA.

In Convergent encryption has been used to enforce data confidentiality. Data copy is encrypted below a key beneath by confusion the data itself. This convergent key is used for encrypt and decrypt a data copy. Moreover, such not permitted users cannot decrypt the cipher text even conspire with the S-CSP (storage cloud service provider). Security analysis make obvious that that system is secure in terms of the description particular in the planned security model.

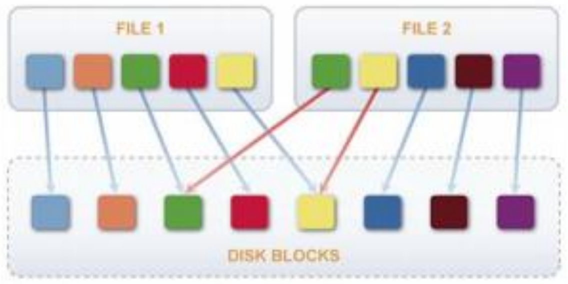

Figure 1: Architecture for Authorized **Deduplication**

This work known a company by where the employee data such as name, password, email id, contact number and designation is registered by admin or owner of the company based on his userid and password employees of the company able to perform operations such as file upload download and duplicate checks on the files based on his privileges.

There are three entities define in hybrid cloud architecture of authorized deduplication.

- **Data Users:** Outsource data storage to the S-CSP(storage cloud service provider) and access the data later. In a storage system supporting deduplication, the user only uploads EXCLUSIVE data but does not upload any duplicate data to save the upload bandwidth, which may be owned by the same user or different users. Each file is confined with the convergent encryption key and privilege keys to understand the authorized deduplication with discrepancy privileges.
- **Private Cloud:** This is new entity for facilitating users secure use of cloud services. The private keys for privileges are managed by private cloud, which provides the file token to users. Specifically, since the computing resources at data user/owner side are controlled and the public cloud is not fully trusted in carry out, private cloud is able to provide data user/owner with an finishing situation and infrastructure working as an interface among user and the public cloud.
- **S-CSP(storage cloud service provider):**This is an entity that provides a data storage service in public cloud. The S- CSP make available the data outsourcing service and stores data in support of the users. To decrease the storage cost, the S- CSP reducing the storage of redundant data via deduplication and keeps only unique data. In this paper, we assume that S-CSP is always online and has abundant storage capacity and computation power.

Algorithm

```
Step 1: Start
Step 2: A User tries to login into the cloud
Step 3: Admin checks the user login id and passwords
Step 4 :If (Login details = = correct)
Step 5: User is login tinto the cloud
Step 6: Else if(Login details = = In correct)
Step 7: Reject that particluar user login.
Step 8: On SucessLogin Upload Data and Control
        Duplicate Check.
Step 9: Stop
```

Algorithm Explanation

Our completion of the **Client** provides the following function calls to support token generation and deduplication along the file upload process.

- FileTag(File) - It computes SHA-1 hash of the File as File Tag;
- TokenReq(Tag, UserID) - It requests the Private Server for File Token generation with the File Tag and User ID;
- DupCheckReq(Token) - It requests the Storage Server for Duplicate Check of the File by sending the file token received from private server;
- ShareTokenReq(Tag, {Priv.}) - It requests the Private Server to generate the Share File Token with the File Tag and Target Sharing Privilege Set;
- FileEncrypt(File) - It encrypts the File with Convergent Encryption using 256-bit AES algorithm in cipher block chaining (CBC) mode, where the convergent key is from SHA-256 Hashing of the file;
- FileUploadReq(FileID, File, Token) – It uploads the File Data to the Storage Server if the file is Unique and updates the
- File Token stored. Our completion of the Private Server includes matching request handlers for the token production and retain a key storage with Hash Map.
- TokenGen(Tag, UserID) - It loads the connected privilege keys of the user and produce the token with HMAC-SHA-1 algorithm

IV. EXPERIMENTAL STUDY

We show the system model for cloud storage auditing with verifiable outsourcing of key updates in Fig. 1. There are three parties in the model: the client, the cloud and the third-party auditor (TPA). The client is the owner of the files that are uploaded to cloud. The total size of these files is not fixed, that is, the client can upload the growing files to cloud in different time points. The cloud stores the client’s files and provides download service for the client. The TPA plays two important roles: the first is to audit the data files stored in cloud for the client; the second is to update the encrypted secret keys of the client in each time period. The TPA can be considered as a party with powerful computational capability or a service in another independent cloud. Similar to [23], the whole lifetime of the files stored in cloud is divided into T + 1 time periods (from 0-th to T -th time periods). Each file is assumed to be divided into multiple blocks. In order to simplify the description, we do not furthermore divide each block into multiple sectors [7] in the description of our protocol. In the end of each time period, the TPA updates the encrypted client's secret key for cloud storage auditing according to the next time period. But the public key keeps unchanged in the whole time periods. The client sends the key requirement to the TPA only when he wants to upload new files to cloud. And then the TPA sends the encrypted secret key to the client. After that, the client decrypts it to get his real secret key, generates authenticators for files, and uploads these files along with authenticators to cloud. In addition, the TPA will audit whether the files in cloud are stored correctly by a challenge-response protocol between it and the cloud at regular time.

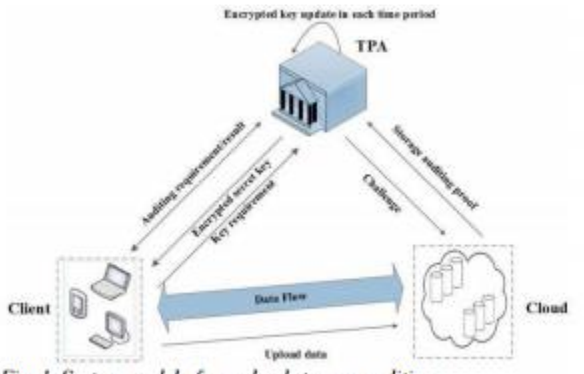

Fig. 1. System model of our cloud storage auditing.

B. Definitions

(1) The definition of cloud storage auditing protocol with verifiable outsourcing of key updates. Definition 1: A cloud storage auditing protocol with secure outsourcing of key updates is composed by seven algorithms (SysSetup, EkeyUpdate, VerESK, DecESK, AuthGen, Proof-Gen, ProofVerify), shown below:

1) SysSetup: the system setup algorithm is run by the client. It takes as input a security parameter k and the total number of time periods T , and generates an encrypted initial client's secret key E S K0, a decryption key D K and a public key P K . Finally, the client holds D K , and sends E S K0 to the TPA.

2) EkeyUpdat e: the encrypted key update algorithm is run by the TPA. It takes as input an encrypted client's secret key E S K j , the current period j and the public key P K , and generates a new encrypted secret key E S K j +1 for period j + 1.

3) VerESK : the encrypted key verifying algorithm is run by the client. It takes as input an encrypted client's secret key E S K j , the current period j and the public key P K , if E S K j is a well-formed encrypted client's secret key, returns 1; otherwise, returns 0.

4) DecESK : the secret key decryption algorithm is run by the client. It takes as input an encrypted client's secret key E S K j , a decryption key D K , the current period j and the public key P K , returns the real client's secret key S K j in this time period.

5) AuthGen: the authenticator generation algorithm is run by the client. It takes as input a file F, a client's secret key S K j , the current period j and the public key P K , and generates the set of authenticators for F in time period j .

6) Proof Gen: the proof generation algorithm is run by the cloud. It takes as input a file F, a set of authenticators , a challenge Chal, a time period j and the public key P K , and generates a proof P which proves the cloud stores F correctly.

7) Proof Verify: the proof verifying algorithm is run by the TPA. It takes as input a proof P, a challenge Chal, a time period j , and the public key P K , and returns "True" if P is valid; or "False", otherwise.

(2) Definition of Security As same as other cloud storage auditing protocols [5]–[7], [9]–[13], [15]–[18], [20], the malicious cloud is viewed as the adversary in our security model. We use three games (Game 1, Game 2 and Game 3) to
describe the adversaries with different compromising abilities who are against the security of the proposed protocol. Specifically, Game 1 describes an adversary, who fully compromises the TPA to get all encrypted secret keys E S K j (periods j = 0, . . . , T), tries to forge a valid authenticator in any time period. This game, in fact, shows the security should satisfy that the TPA cannot help the cloud to forge any authenticator in any time period even if it knows the encrypted secret keys. Game 2 describes an adversary, who compromises the client to get D K ,

tries to forge a valid authenticator in any time period. This game, in fact, shows the security should satisfy that an adversary cannot forge any authenticator in any time period even if it gets the decryption secret key D K by attacking the client. Game 3 provides the adversary more abilities, which describes an adversary, who compromises the client and the TPA to get both E S K j and D K at one time period j , tries to forge a valid authenticator before time period j . This game, in fact, shows the security should satisfy that an adversary cannot forge any authenticator prior to one certain time period if it attacks the TPA and the client simultaneously to get their secret keys in this time period

Our evaluation focuses on comparing the overhead induced by authorization steps, including file token generation and share token generation, beside the convergent encryption and file upload steps. We appraise the overhead by unreliable various factors, together with 1) File Size 2) Number of Stored Files 3) Deduplication Ratio 4) Privilege Set Size. We break down the upload process into 6 steps, 1) Tagging 2) Token Generation 3) Duplicate Check 4) Share Token Generation 5) Encryption 6) Transfer . For each step, we record the start and end time of it

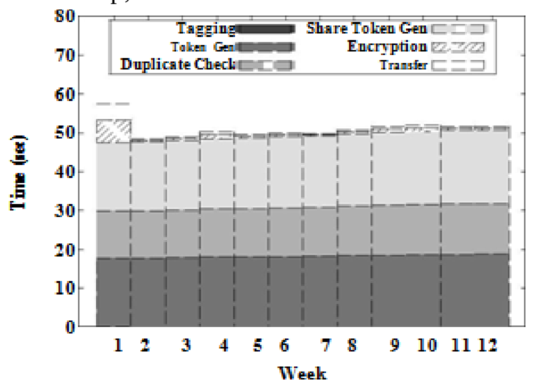

Fig. 6. Time Breakdown for the VM

Fig Time Breakdown for the VM

and therefore obtain the breakdown of the total time spent. We present the regular time taken in every data set in the figures

File Size

To appraise the consequence of file size to the time spent on various steps, we upload 100 unique files (i.e., without any deduplication opportunity) of particular file size and record the time break down. Using the unique files enables us to evaluate the worst-case scenario where we have to upload all file data. The average time of the steps from test sets of different file size are plotted in Figure 2. The time spent on tagging, encryption, upload enlarge linearly with the file size, since these operations involve the actual file data and incur file I/O with the whole file.

Number of Stored Files

To evaluate the effect of number of stored files in the system, we upload 10000 10MB unique files to the system and record the breakdown for every file upload. From Figure 3, every step remains constant along the time. Token checking is done with a hash table and a linear search would be carried out in case of collision.

To appraise the consequence of the deduplication ratio, we prepare two unique data sets, each of which consists of 50 100MB files. We first upload the first set as an initial upload. For the second upload, we pick a portion of 50 files, through given deduplication ratio from the initial set as duplicate files and remaining files from the second set as unique files. The average time of uploading the second set is presented in Figure 2.

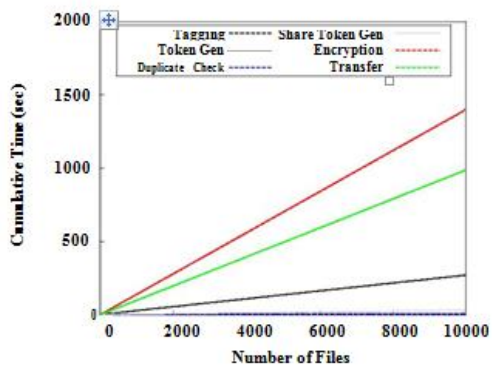

Fig 2. Time Breakdown for different Number of stored files

To evaluate the effect of privilege set size, we upload 100 10MB unique files with different size of the data owner and target share privilege set size. In Figure 5, it shows by taking token generation increases linearly as more keys are associated with the file and also the duplicate check time. While the number of keys increases 100 times from 1000 to 100000, the total time spent only increases to 3.81 times and it is noted that the file size of the experiment is set at a small level (10MB), the effect would become less significant in case of larger files.

Conclusion and Future

In this proposed architecture we have designed a new notion for removing data deduplication and to protect the data security through privileges of users and duplicate check. We had perform various new deduplication constructions behind authorized duplicate check in hybrid cloud architecture, in which the duplicate-check tokens of files are produced by the private cloud server with private keys. As the notion in this project we realize a prototype of our considered authorized duplicate check scheme and conduct test bed experiments on our prototype. From this project we show that our sanctioned duplicate check scheme acquire negligible overhead balance to convergent encryption and network relocate.

VI. CONCLUSION

In this paper, we concentrate on the best way to make the key overhauls as straightforward as could be expected under the circumstances for the customer and propose another worldview called distributed storage reviewing with certain outsourcing of key redesigns. In this worldview key overhauls can be securely outsourced to some approved gathering and along these lines the key-upgrade trouble on the customer will be kept insignificant. Inparticular, we influence the outsider inspector (TPA) in numerous current open examining outline, let it assume the part of approved gathering for our situation and make it accountable for both the capacity reviewing and secure key upgrades for key-presentation resistance. As of late, key presentation issue in the settings of distributed storage examining has been proposed and concentrated on. In this worldview, key redesigns can be securely outsourced to some approved gathering, and subsequently the key-overhaul load on the customer will be kept insignificant. In particular, we influence the outsider evaluator (TPA) in numerous current open examining plans, let it assume the part of approved gathering for our situation, and make it accountable for both the capacity inspecting and the safe key upgrades for key-introduction resistance. Moreover, our plan additionally outfits the customer with capacity to facilitate confirm the legitimacy of the scrambled mystery keys gave by TPA. We formalize the definition and the security model of this worldview. while the client can further verify the validity of the encrypted secret keys when downloading them from the TPA. We give the formal security proof and the performance simulation of the proposed scheme.The security confirmation and the execution reenactment demonstrate that our point by point plan instantiations are secure and productive.

REFERENCE

[1] [1] M. J. Atallah, K. N. Pantazopoulos, J. R. Rice, and E. E. Spafford, “Secure outsourcing of scientific computations,” Adv. Comput., vol. 54,pp. 215–272, 2002.

[2] D. Benjamin and M. J. Atallah, “Private and cheating-free outsourcing of algebraic computations,” in Proc. 6th Annu Conf. Privacy, Secur. Trust, 2008, pp. 240–245.

[3] C. Wang, K. Ren, and J. Wang, “Secure and practical outsourcing of linear programming in cloud computing,” in Proc. IEEE INFOCOM, Apr. 2011, pp. 820–828.

[4] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, “New algorithms for secure outsourcing of modular exponentiations,” in Proc. 17th Eur. Symp. Res. Comput. Secur., 2012, pp. 541–556.

[5] G. Ateniese et al., “Provable data possession at untrusted stores,” in Proc. 14th ACM Conf. Comput. Commun. Secur., 2007, pp. 598–609.

[6] A. Juels and B. S. Kaliski, Jr., “PORs: Proofs of retrievability for large files,” in Proc. 14th ACM Conf. Comput. Commun. Secur., 2007,pp. 584–597.

[7] H. Shacham and B. Waters, “Compact proofs of retrievability,” in Advances in Cryptology. Berlin, Germany: Springer-Verlag, 2008,pp. 90–107.

[8] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, “Scalable and efficient provable data possession,” in Proc. 4th Int. Conf. Secur. Privacy Commun. Netw., 2008, Art. ID 9.

[9] F. Sebe, J. Domingo-Ferrer, A. Martinez-balleste, Y. Deswarte, and J. Quisquater, “Efficient remote data possession checking in critical information infrastructures, IEEE Trans. Knowl. Data Eng., vol. 20, no. 8, pp. 1034–1038, Aug. 2008.

[10] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, “MR-PDP: Multiple-replica provable data possession,” in Proc. 28th IEEE Int. Conf. Distrib. Comput. Syst., Jun. 2008, pp. 411–420.

[11] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Efficient provable data possession for hybrid clouds," in Proc. 17th ACM Conf. Comput. Commun. Secur., 2010, pp. 756–758.
[12] C. Wang, K. Ren, W. Lou, and J. Li, "Toward publicly auditable secure cloud data storage services," IEEE Netw., vol. 24, no. 4, pp. 19–24, Jul./Aug. 2010.
[13] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud comput-ing," IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5, pp. 847–859, May 2011.
[14] K. Yang and X. Jia, "Data storage auditing service in cloud computing: Challenges, methods and opportunities," World Wide Web, vol. 15, no. 4,pp. 409–428, 2012