Enabling Identity-Based Integrity Auditing and Data Sharing with Sensitive Information Hiding for Secure Cloud Storage

Prof. T. S. Pawar[1], Prof. T. P. Chavan[2], Prof. N. T. Sabale[3]
[1,2,3]*Member, S. P. I. T. Polytechnic, Kurund*

Abstract: **In fact, visitors can use cloud storage services to store their data remotely and share it with others. To ensure the integrity of the data saved in the cloud, remote data integrity auditing is suggested. The cloud file may contain sensitive data in some popular cloud storage systems, such the electronic health records system. When the cloud file is shared, the private data shouldn't be made public. Sensitive information can be hidden by encrypting the entire shared file, but doing so will prevent others from using it. To date, there has been no investigation on how to do data sharing while concealing sensitive information in remote data integrity audits. In this research, we offer a remote data integrity auditing approach that addresses this issue by enabling data sharing while concealing critical information. In this technique, the data blocks that contain the sensitive information of the file are sanitized using a sanitizer, which also changes the signatures of these data blocks to ones that are valid for the sanitized file. In the integrity auditing process, these signatures are used to confirm the sanitized file's integrity. Therefore, our plan allows the cloud-stored file to be shared and utilized by others as long as the sensitive information is concealed, allowing for the fast execution of remote data integrity audits.**

Index Terms**: data integrity auditing, Cloud storage, data sharing, sensitive information hiding.**

I. INTRODUCTION

Members find it extremely difficult to keep the vast amount of data locally due to its rapid expansion. As a result, an increasing number of businesses and people want to keep their data on the cloud. However, because of the unavoidable hardware malfunctions, software defects, and human mistake in the cloud, the data saved there may be lost or corrupted. Numerous remote data integrity auditing techniques have been put out to confirm whether the data is stored accurately in the cloud.

Before uploading data blocks to the cloud, the data owner must first create signatures for them in remote data integrity auditing methods. During the integrity auditing phase, these signatures are used to demonstrate that the cloud actually owns these data blocks. These data blocks, along with the appropriate signatures, are subsequently uploaded to the cloud by the data owner. Many cloud storage apps, including Google Drive, Dropbox, and iCloud, frequently allow multiple users to share data stored in the cloud. One of the most popular advantages of cloud storage is data sharing, which enables many users to share their data with other people. However, some sensitive information may be included in these cloud-stored shared data. The Electronic Health Records (EHRs) stored and shared in the cloud usually contain patients' sensitive information (patient's name, telephone number and ID num- ber, etc.) and the hospital's sensitive information (hospital's name, etc.). If these EHRs are directly uploaded to the cloud to be shared for research purposes, the sensitive information of patient and hospital will be inevitably exposed to the cloud and the researchers. Besides, the integrity of the EHRs needs to be guaranteed due to the existence of human errors and software/hardware failures in the cloud. Therefore, it is important to accomplish remote data integrity auditing on the condition that the sensitive information of shared data is protected.

This encrypted file, then upload it to the cloud along with the associated signatures. Since this file can only be decrypted by the data owner, this strategy can actually hide sensitive information. It will, however, prevent others from using the entire shared file. For instance, encrypting the electronic health records of patients with infectious diseases helps safeguard patient and hospital privacy; but, researchers are no longer able to use these encrypted EHRs efficiently. Giving the researchers the decryption key appears to

be one way to address the aforementioned issue.

However, for the reasons listed below, this approach is not viable to implement in practical situations. First, the distribution of the decryption key requires safe methods, which can be challenging in some situations. Furthermore, when a user uploads their EHRs to the cloud, it appears to be very impossible to determine which researchers may use them in the near future. Therefore, it is not feasible to encrypt the entire shared file in order to conceal important information. Therefore, it is crucial and helpful to understand how to implement data sharing while concealing sensitive information in remote data integrity audits. Regretfully, prior study has not examined this issue.

A. AN ILLUSTRATIVE EXAMPLE FOR EHRS

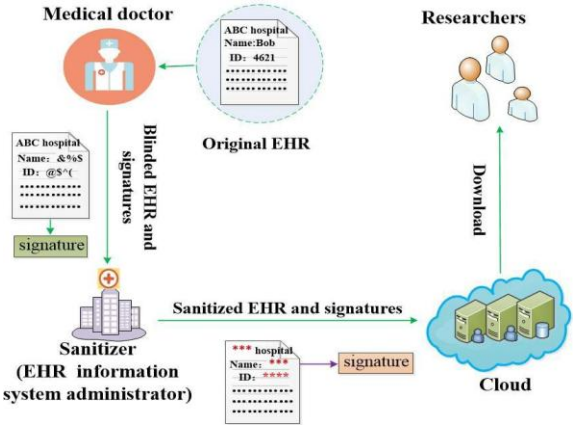

Fig. 1. Example of EHRs

Here, we provide an example of an EHR in Fig. 1. There are two components to the sensitive data in EHRs in this scenario. Names and ID numbers are examples of personal sensitive information, often known as patient's sensitive information. The other is sensitive information about the organization (sensitive information about the hospital), including the hospital's name.B. Final Stage After your paper has been accepted. The authors of the accepted manuscripts will be given a copyright form and the form should accompany your final submission. In general, when EHRs are uploaded to the cloud for research purposes, the sensitive information mentioned above should be substituted with wildcards. One way to think of the sanitizer is as the hospital's EHR information system administrator. The sanitizer shouldn't come into contact with sensitive personal data. Furthermore, the cloud and shared users shouldn't have access to any sensitive data. In order to store patient EHRs in the EHR information system, a doctor must create and send them to the sanitizer. However, these EHRs typically include sensitive patient and hospital data, like the patient's name, ID number, and hospital name. In order to protect patient privacy from the sanitizer, the doctor will blind all sensitive patient data in each EHR before sending it to the sanitizer. After creating signatures for this blindfolded EHR, the doctor forwards them to the sanitizer.

These messages are stored in the EHR information system by the sanitizer. The doctor sends a request to the sanitizer when he requires the EHR. The sanitizer then sends the blinded EHR to the doctor after downloading it from the EHR information system. Ultimately, from this blinded EHR, the physician retrieves the original EHR.

In order to standardize the format when this EHR is uploaded and shared in the cloud for research purposes, the sanitizer must clean the data blocks that relate to the patient's private data. Additionally, the sanitizer must sanitize the data blocks that match to the hospital's sensitive information in order to maintain hospital privacy.

Usually, wildcards are used in place of these data chunks. Additionally, the sanitizer has the ability to change the signatures on these data blocks into ones that are legitimate for the cleaned EHR. It enables the efficient execution of remote data integrity auditing. The sanitizer does not have to come into contact with medical professionals during sanitizing. These sanitized EHRs and the associated signatures are then uploaded to the cloud by the sanitizer. Researchers can share and use the EHRs in this fashion, and the sensitive data within them can be concealed. In the interim, it is possible to guarantee the integrity of these cloud-based EHRs.

The following factors make the sanitizer essential. First of all, the contents of the data blocks containing the patient's sensitive information may become jumbled code once they have been blinded. By substituting the contents of these data blocks with wildcards, the sanitizer can standardize the format. In order to safeguard the hospital's privacy, the sanitizer can additionally use wildcards to sanitize the data blocks that relate to sensitive hospital information, including the hospital's name. Second, the information management can be made easier by the sanitizer. It can sanitize the EHRs in bulk, and uploads these sanitized

EHRs to the cloud at a fixed time. Thirdly, when the medical doctor needs the EHR, the sanitizer as the administrator of EHR information system can download the blinded EHR from the EHR information system and sends it to the medical doctor. The medical doctor can recover the original EHR from the blinded one.

B. RELATED WORK

Several remote data integrity auditing techniques have been put out to confirm the accuracy of the data kept in the cloud. A Third Party Auditor (TPA) is added to periodically check the integrity of the cloud data on behalf of the user, hence reducing the computation burden on the user's end. Ateniese and associates. [2] To guarantee data possession on the untrusted cloud, the concept of Provable Data Possession (PDP) was originally put out. To accomplish blockless verification and lower I/O costs, their suggested approach makes use of homomorphic authenticators and random sampling techniques. Kaliski and Juels [3] offered a workable plan and established a concept known as Proof of Retrievability (PoR). This plan makes it possible to recover and guarantee the integrity of data stored in the cloud. Shacham and Waters [4] suggested a private remote data integrity auditing technique and a public remote data integrity auditing scheme based on the pseudorandom function and BLS signature.

In order to protect the data privacy, Wang et al. [5] proposed a privacy-preserving remote data integrity auditing scheme with the employment of a random masking technique. Worku et al.[6]. Utilized a different random masking technique to further construct a remote data integrity auditing scheme supporting data privacy protection. This scheme achieves better efficiency compared with the scheme in [5]. In order to lessen the user's computational load associated with creating signatures, Guan et al.[7] created an auditing system for remote data integrity based on the indistinguishability obfuscation technique. Shen and associates.[8]. introduced a Third Party Medium (TPM) to design a light-weight remote data integrity auditing scheme. In this scheme, the TPM helps user generate signatures on the condition that data privacy can be protected. In order to support data dynamics, Ateniese et al.[10]. firstly proposed a partially dynamic PDP scheme. Erway et al. [11] used a skip list to construct a fully data dynamic auditing scheme. Wang et al. [12] proposed another remote data integrity auditing scheme supporting full data dynamics by utilizing Merkle Hash Tree. To reduce the damage of users' key exposure, Yu et al. [13] and [14], and Yu and Wang [15] proposed key-exposure resilient remote data integrity auditing schemes based on key update technique [16].

The data sharing is an important application in cloud storage scenarios. To protect the identity privacy of user, Wang *et al.* [17] designed a privacy-preserving shared data integrity auditing scheme by modifying the ring signature for secure cloud storage. Yang *et al.* [18] constructed an efficient shared data integrity auditing scheme, which not only supports the identity privacy but only achieves the identity traceability of users. Fu *et al.* [19] designed a privacy-aware shared data integrity auditing scheme by exploiting a homomorphic verifiable group signature. In order to support efficient user revocation, Wang *et al.* [20] proposed a shared data integrity auditing scheme with user revocation by using the proxy re-signature. With the employment of the Shamir secret shar- ing technique, Luo *et al.* [21] constructed a shared data integrity auditing scheme supporting user revocation. The aforementioned schemes all rely on Public Key Infrastructure (PKI), which incurs the considerable overheads from the complicated certificate management. To simplify certificate management, Wang [22] proposed an identity-based remote data integrity auditing scheme in multicloud storage. This scheme used the user's identity information such as user's name or e-mail address to replace the public key.

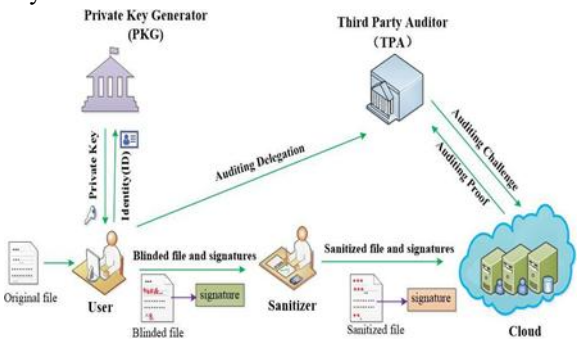

Fig. 2. The system model

cryptosystems. Wang et al. [25] proposed an identity-based data integrity auditing scheme satisfying unconditional anonymity and incentive. Zhang et al. [26] proposed an identity-based remote data integrity auditing scheme for shared data support- ing real

efficient user revocation.

Other aspects, such as privacy-preserving authenticators [27] and data deduplication [28], [29] in remote data integrity auditing have also been explored. However, all of existing remote data integrity auditing schemes cannot support data sharing with sensitive information hiding. In this paper, we explore how to achieve data sharing with sensitive information hiding in identity-based integrity auditing for secure cloud storage.

II.SYSTEM MODEL AND SECURITY MODEL

A. System Model

The system model involves five kinds of different entities: the cloud, the user, the sanitizer, the Private Key Generator (PKG) and the Third Party Auditor (TPA), as shown in Fig.2.

(1) Cloud: The cloud provides enormous data storage space to the user. Through the cloud storage service, users can upload their data to the cloud and share their data with others.

(2) User: The user is a member of an organization, which has a large number of files to be stored in the cloud.

(3) Sanitizer: The sanitizer is in charge of sanitizing the data blocks corresponding to the sensitive information (personal sensitive information and the organization's sensitive informa- tion) in the file, transforming these data blocks' signatures into valid ones for the sanitized file, and uploading the sanitized file and its corresponding signatures to the cloud.

(4) PKG: The PKG is trusted by other entities. It is responsible for generating system public parameters and the private key for the user according to his identity *ID*.

(5) TPA: The TPA is a public verifier. It is in charge of verifying the integrity of the data stored in the cloud on behalf of users.

In order to generate the appropriate signatures, the user first blinds the data blocks that match to the file's sensitive personal information. These signatures serve to confirm the file's integrity and guarantee its authenticity. The blinded file and the associated signatures are then sent to the sanitizer by the user. Following receipt of the user's message, the sanitizer sanitizes these blinded data blocks as well as the data blocks that include sensitive information about the organization. It then converts the sanitized data blocks' signatures into ones that are legitimate for the sanitized file.

This sanitized file and the associated signatures are then sent to the cloud by the sanitizer. During the integrity auditing process, these signatures are used to confirm the sanitized file's integrity. The TPA sends an auditing challenge to the cloud in order to confirm the integrity of the sanitized file that is stored there. The cloud then provides an auditing proof of data possession in response to the TPA. Lastly, the TPA checks to see if this auditing proof is accurate in order to confirm the integrity of the sanitized file.

B. Design Goals

To efficiently support data sharing with sensitive information hiding in identity-based integrity auditing for secure cloud storage, our scheme is designed to achieve the following goals:

1) The correctness:
   a. Private key correctness: to ensure that when the PKG sends a correct private key to the user, this private key can pass the verification of the user.
   b. The correctness of the blinded file and its corresponding signatures: to guarantee that when the user sends a blinded file and its corresponding valid signatures to the sanitizer, the blinded file and its corresponding signatures he generates can pass the verification of the sanitizer.
   c. Auditing correctness: to ensure that when the cloud properly stores the user's sanitized data, the proof it generates can pass the verification of the TPA.
2) Sensitive information hiding: to make sure that none of the file's sensitive information is visible to the cloud or other shared users, and that none of the file's private sensitive information is accessible to the sanitizer.
3) Auditing soundness: to ensure that the cloud cannot pass the TPA's verification if it does not actually hold users' intact, cleansed data.

C. Definition

The following six algorithms make up an identity-based shared data integrity auditing technique that

conceals critical information for safe cloud storage. *Setup, Extract, SigGen, Sanitization, ProofGen and ProofVerif y. Specifically, these algorithms are described as follows:*

1) Setup(1^k) is a setup algorithm run by the PKG. It takes as input a security parameter *k*. It outputs the master secret key *msk* and the system public parameters *pp*.
2) Extract (pp, msk, ID) is an extraction algorithm run by the PKG. It takes as input the system public parameters *pp*, the master secret key *msk*, and a user's identity *ID*. It outputs the user's private key sk_{ID}. The user can verify the correctness of sk_{ID} and accept it as his private key only if it passes the verification.
3) SigGen $(F, sk_{ID}, ssk, name)$ is a signature generation algorithm run by the user *ID*. It takes as input the original file *F*, the user's private key sk_{ID}, the user's signing private key *ssk* and the file identifier name *name.*

   It outputs a blinded file F^*, its corresponding signature set Φ and a file tag c.
4) Sanitization (F^*, Φ) is a sensitive information sanitiza- tion algorithm run by the sanitizer. It takes as input the blinded file F^* and its signature set Φ. It outputs the sanitized file F^r and its corresponding signature set Φ^r.
5) ProofGen$(F^r, \Phi^r, chal)$ is a proof generation algorithm run by the cloud. It takes as input the sanitized file F^r, the corresponding signature set Φ^r and the auditing challenge *chal*. It outputs an auditing proof *P* that is used to demonstrate the cloud truly possesses this sanitized file F^r.
6) ProofVerify $(chal, pp, P)$ is a proof verification algorithm run by the TPA. It takes as input the auditing challenge *chal*, the system public parameters *pp* and the auditing proof *P*. The TPA can verify the correctness of proof *P* the PKG. It takes as input the system public parameters *pp*, the master secret key *msk*, and a user's identity *ID*. It outputs the user's private key sk_{ID}. The user can verify the correctness of sk_{ID} and accept it as his private key only if it passes the verification.
7) SigGen $(F, sk_{ID}, ssk, name)$ is a signature generation algorithm run by the user *ID*. It takes as input the original file *F*, the user's private key sk_{ID}, the user's signing private key *ssk* and the file identifier name *name.*

   It outputs a blinded file F^*, its corresponding signature set Φ and a file tag $c.$
8) Sanitization (F^*, Φ) is a sensitive information sanitiza- tion algorithm run by the sanitizer. It takes as input the blinded file F^* and its signature set Φ. It outputs the sanitized file F^r and its corresponding signature set Φ^r.
9) ProofGen$(F^r, \Phi^r, chal)$ is a proof generation algorithm run by the cloud. It takes as input the sanitized file F^r, the corresponding signature set Φ^r and the auditing challenge *chal*. It outputs an auditing proof *P* that is used to demonstrate the cloud truly possesses this sanitized file F^r.
10) ProofVerify $(chal, pp, P)$ is a proof verification algo- rithm run by the TPA. It takes as input the auditing challenge *chal*, the system public parameters *pp* and the auditing proof *P*. The TPA can verify the correctness of proof *P*.

D. Security Model

We illustrate how adversary A is against the security of an identity-based shared data integrity auditing method with sensitive information concealment by using a game between a challenger C and an adversary A to formalize the security model. According to our security paradigm, the untrusted cloud server is seen as an adversary A and the data owner as a challenger C. The phases of this game are as follows:

1) Setup phase. The challenger *C* runs the *Setup* algorithm to obtain the master secret key *msk* and the system public parameters *pp*, and then sends the public parameters *pp* to the adversary *A*.
2) Query phase. In this phase, the adversary *A* makes the following two queries to the challenger *C*.

   a) *Extract Queries*: The adversary *A* queries the private key for the identity *ID*. The challenger *C runs the Extract algorithm to generate the private key skID, and sends it to the adversary A.*

   b) *SigGen Queries*: The adversary *A* queries the signatures of the file *F*. By running the *Extract*

algorithm, the challenger C gets the private key. And then the challenger C runs the *SigGen* algorithm to calculate the signatures of the file F. Finally, the challenger C sends these signatures to the adversary A.

3) Challenge phase. In this phase, the adversary A acts as a prover and the challenger C plays the role of a verifier. The challenger C sends the challenge $chal = \{i, v_i\}_{i \in I}$ to the adversary A, where $I \in \{\gamma_1, \gamma_2, \ldots, \gamma_c\}$ ($\gamma_j \in [1, n], j \in [1, c]$ and $c \in [1, n]$). Meanwhile, it requests the adversary A to provide a data possession proof P for the data blocks $\{m_{\gamma_1}, m_{\gamma_2}, \ldots, m_{\gamma_c}\}$ under the *chal*.
4) Forgery phase. After receiving the challenge from the challenger C, the adversary A generates a data possession proof P for the data blocks indicated by *chal* to reply the challenger C. If this proof P can pass the verification of the challenger C with non-negligible probability, we say that the adversary A succeeds in the above game.

In the above security model, we need to prove that if an adversary A, who does not keep all the data blocks challenged by the challenger C, cannot guess all the corrupted data blocks, then it cannot generate a valid proof P to pass the verification of the challenger C. The goal of the adversary A is to pass the verification of the challenger C by generating a valid proof P for the challenged data blocks. The definition 2 presents that there exists a knowledge extractor that can capture the challenged data blocks whenever the adversary can output a valid data possession proof P. The definition 3 is to describe the detectability for the data integrity auditing scheme, which can ensure that the cloud truly keeps the data blocks that are not challenged with high probability

Definition 2: We say a remote data integrity auditing scheme is secure if the following condition holds: whenever an adversary A in the aforementioned game can generate a valid proof P to pass the verification of the challenger C with non-negligible probability, there exists a knowledge extractor that can capture the challenged data blocks except possibly with negligible probability.

Definition 3: A remote data integrity auditing scheme is (ρ, δ) $(0 < \rho, \delta < 1)$ detectable if the cloud corrupted ρ fraction of the whole file, these corrupted data blocks can be detected with the probability at least δ.

We consider the sanitizer is not fully trustworthy. The sanitizer might be curious about the personal sensitive information of the file. In addition, the cloud and the shared users might be curious about all of the sensitive information of the file. Thus, the personal sensitive information of the file should not be exposed to the sanitizer, and all of the sensitive information of the file should not be exposed to the cloud and the shared users in our scheme. That is to say, even if the sanitizer is untrustworthy, the personal sensitive information of the file will not be exposed to it. Furthermore, even if the cloud and the shared users are untrustworthy, all of the sensitive

Definition 4: We say a remote data integrity auditing scheme achieves sensitive information security if the sanitizer cannot derive the personal sensitive information of the file, besides the cloud and the shared users cannot derive any sensitive information of the file.

III THE PROPOSED SCHEME

A. An Overview

We examine using the concept of the sanitizable signature to accomplish data sharing while concealing sensitive information. to sanitize the sensitive information of the file by introducing an authorized sanitizer. Nonetheless, it is infeasible if this sanitizable signature is directly used in remote data integrity auditing. Firstly, this signature in is constructed based on chameleon hashes However, a lot of chameleon hashes exhibit the key exposure problem. To avoid this security problem, the signature used in [30] requires strongly unforgeable chameleon hashes, which will inevitable incur huge computation overhead [31]. Secondly, the signature used in [30] does not support blockless verifiability. It means that the verifier has to download the entire data from the cloud to verify the integrity of data, which will incur huge communication overhead and excessive verification time in big data storage scenario. Thirdly, the signature used in [30] is based on the PKI, which suffers from the complicated certificate management.

We create a new, effective signature algorithm in the signature generation step to solve the aforementioned issues. Blockless verifiability, which enables the verifier to examine the integrity of data without downloading the complete set from the cloud, is supported by the signature scheme that was devised. Furthermore, its foundation is identity-based cryptography, which streamlines the intricate certificate administration process. In our proposed scheme, the PKG generates the private key for user according to his identity *ID*. The user can check the correctness of the received private key. When there is a desire for the user to upload data to the cloud, in order to preserve the personal sensitive information of the original file from the sanitizer, this user needs to use a blinding factor to blind the data blocks corresponding to the personal sensitive information of the original file. When necessary, the user can recover the original file from the blinded one by using this blinding factor. And then this user employs the designed signature algorithm to generate signatures for the blinded file. These signatures will be used to verify the integrity of this blinded file. In addition, the user generates a file tag, which is used to ensure the correctness of the file identifier name and some verification values. The user also computes a transformation value that is used to transform signatures for sanitizer. Finally, the user sends the blinded file, its corresponding signatures, and the file tag along with the transformation value to the sanitizer. When the above messages from user are valid, the sanitizer firstly sanitizes the blinded data blocks into a uniform format and also sanitizes the data blocks corresponding to the organization's sensitive information to protect the privacy of organization, and then transforms their corresponding signatures into valid ones for sanitized file using transformation value. Finally, the sanitizer uploads the sanitized file and the corresponding signatures to the cloud. When the data integrity auditing task is performed, the cloud generates an auditing proof according to the challenge from the TPA. The TPA can verify the integrity of the sanitized file stored the cloud by checking whether this auditing proof is correct or not. The details will be described in the following subsection.

IV. CONCLUSION

In this paper, we proposed an identity-based data integrity auditing scheme for secure cloud storage, which supports data sharing with sensitive information hiding. In our scheme, the file stored in the cloud can be shared and used by others on the condition that the sensitive information of the file is protected The security proof and the experimental analysis demonstrate that the proposed scheme achieves desirable security and efficiency.

REFERENCE

[1] Wenting Shen, Jing Qin, Jia Yu. Rong Hao, and Jiankun Hu" Enabling Identity-Based Integrity Auditing and Data Sharing with Sensitive Information Hiding for Secure Cloud Storage" IEEE Transactions on Information Forensics and Security, Vol. 14, No. 2, February 2019

[2] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan. 2012.

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

[4] 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.

[5] H. Shacham and B. Waters, "Compact proofs of retrievability," *J. Cryp- tol.*, vol. 26, no. 3, pp. 442–483, Jul. 2013.

[6] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy- preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.

[7] S. G. Worku, C. Xu, J. Zhao, and X. He, "Secure and efficient privacy- preserving public auditing scheme for cloud storage," *Comput. Electr. Eng.*, vol. 40, no. 5, pp. 1703–1713, 2014.

[8] C. Guan, K. Ren, F. Zhang, F. Kerschbaum, and J. Yu, "Symmetric- key based proofs of retrievability supporting public verification," in *Computer Security—ESORICS*. Cham, Switzerland: Springer, 2015, pp. 203–223.

[9] W. Shen, J. Yu, H. Xia, H. Zhang, X. Lu, and R. Hao, "Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium," *J. Netw. Comput. Appl.*, vol. 82, pp. 56–64, Mar. 2017.

[10] J. Sun and Y. Fang, "Cross-domain data sharing in distributed electronic health record

systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 6, pp. 754–764, Jun. 2010.

[11] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netw.*, 2008, Art. no. 9.

[12] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 213–222.

[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] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1167–1179, Jun. 2015.

[15] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1362–1375, Jun. 2016.

[16] J. Yu and H. Wang, "Strong key-exposure resilient auditing for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1931–1940, Aug. 2017.

[17] J. Yu, R. Hao, H. Xia, H. Zhang, X. Cheng, and F. Kong, "Intrusion- resilient identity-based signatures: Concrete scheme in the standard model and generic construction," *Inf. Sci.*, vols. 442–443, pp. 158–172, May 2018.

[18] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," in *Proc. IEEE 5th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2012, pp. 295–302.

[19] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *J. Syst. Softw.*, vol. 113, pp. 130–139, Mar. 2016.

[20] A. Fu, S. Yu, Y. Zhang, H. Wang, and C. Huang, "NPP: A new privacy-aware public auditing scheme for cloud data sharing with group users," *IEEE Trans. Big Data*, to be published, doi: 10.1109/TBDATA.2017.2701347.

[21] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Trans. Serv. Comput.*, vol. 8, no. 1, pp. 92–106, Jan./Feb. 2015.

[22] Y. Luo, M. Xu, S. Fu, D. Wang, and J. Deng, "Efficient integrity auditing for shared data in the cloud with secure user revocation," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2015, pp. 434–442.

[23] H. Wang, "Identity-based distributed provable data possession in multi- cloud storage," *IEEE Trans. Serv. Comput.*, vol. 8, no. 2, pp. 328–340, Mar./Apr. 2015.

[24] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1165–1176, Jun. 2016.

[25] Y. Yu *et al.*, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 767–778, Apr. 2017.

[26] H. Wang, D. He, J. Yu, and Z. Wang, "Incentive and unconditionally anonymous identity-based public provable data possession," *IEEE Trans. Serv. Comput.*, to be published, doi: 10.1109/TSC.2016.2633260.

[27] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Trans. Depend. Sec. Comput.*, to be published, doi: 10.1109/TDSC.2018.2829880.

[28] W. Shen, G. Yang, J. Yu, H. Zhang, F. Kong, and R. Hao, "Remote data possession checking with privacy-preserving authenticators for cloud storage," *Future Gener. Comput. Syst.*, vol. 76, pp. 136–145, Nov. 2017.

[29] J. Li, J. Li, D. Xie, and Z. Cai, "Secure auditing and deduplicating data in cloud," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2386–2396, Aug. 2016.

[30] J. Hur, D. Koo, Y. Shin, and K. Kang, "Secure data deduplication with dynamic ownership management in cloud storage," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 11, pp. 3113–3125, Nov. 2016.

[31] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik, "Sanitizable signatures," in *Proc. 10th Eur. Symp. Res. Comput. Secur.* Berlin,

Germany: Springer-Verlag, 2005, pp. 159–177.
[32] G. Ateniese and B. de Medeiros, "On the key exposure problem in chameleon hashes," in *Security in Communication Networks*. Berlin, Germany: Springer, 2005, pp. 165–179.