Journal of Computers Vol. 28, No. 4, 2017, pp. 23-30
doi:10.3966/199115592017082804003

Security Analysis of Delegable and Proxy Provable Data Possession in Public Cloud Storage

Yong-Jun Ren[1,2], Jian Shen[1,2], Jin Wang[3*], Li-Ming Fang[4]

[1] Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology
Nanjing, P.R.China

[2] School of Computer and Software, Nanjing University of Information Science and Technology
Nanjing, P.R.China
{renyj100,s_shenjian}@126.com

[3] School of Information Engineering, Yangzhou University
Yangzhou, P.R.China
jinwang@yzu.edu.cn

[4] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics
Nanjing, P.R.China
Fangliming@nuaa.edu.cn

Received 30 June 2015; Revised 19 July 2016; Accepted 28 December 2016

Abstract. Cloud storage is now an important development trend in information technology. However, information security has become an important problem to impede it for commercial application, such as data con-fidentiality, integrity, and availability. In this paper, we revisit the two private PDP schemes. We show that the property of correctness cannot be achieved when active adversaries are involved in these auditing systems. More specifically, an active adversary can arbitrarily tamper the cloud data and produce a valid auditing. Moreover, the malicious CSS can put the auditing right to entrust to anyone and control the delegation key and lead to the failure of the subsequent validation work. Finally, we propose a solution using key agreement to resolve the weakness. Our scheme is high efficiency due to removing expensive bilinear computing. Moreover, the verifier is stateless and independent from cloud storage server, which is an important secure property in PDP schemes.

Keywords: cloud storage, provable data possession, proxy signature

1 Introduction

Cloud Computing has been envisioned as the next generation architecture of IT Enterprise, which is defined as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. A recent survey indicates that more than 79% of organizations attempt to utilize data outsourcing because it relieves the burden of maintenance cost as well as the overhead of storing data locally. Moreover, the users are able to access information from anywhere and at any time instead of having to dedicated machines.

Although cloud computing offers several advantages for users, there are some security concerns that prevent a full adoption of the new technology. When users outsource data files in a distant server, the physical access to the file is actually lost and the administration of files is delegated to a cloud provider

* Corresponding Author

as an unreliable third party. The data in the cloud space is also susceptible to various kinds of inside and outside threats that might risk the integrity, confidentiality, and availability of data. Recently, various companies reported data corruption in servers with major cloud infrastructure providers, and many events of cloud service outages, such as, Amazon S3 breakdown and Gmail mass deletion.

After outsourcing the data to the remote clouds, the cloud users must ensure that the data remains intact. However, the conventional integrity verification methods, for example, hash functions and signatures are inapplicable in cloud computing because of the lack of a local data copy. On the other hand, downloading of possibly a large size file is impractical. The aforementioned situation worsens when users are accessing data using mobile devices. The cloud users are responsible to devise a suitable audit mechanism that can remotely verify the intactness of distant data.

In order to solve the problem of data auditing service, researchers proposed many schemes under different systems and security models. Considering the role of the PDP verifier, the PDP protocols can be classified into two categories: private PDP and public PDP. In the response checking phase of private PDP, some private information is needed. While in the response checking of public PDP, the private information is not needed. Private PDP is necessary in some cases. Recently Shen and Tzeng presented delegable provable data possession scheme [19], in which data owner generates the delegation key for delegated verifier and store the key in CSSs for verification. Wang also proposed a proxy provable data possession (PPDP) model [20] and provided a construction for it. In PPDP data owner can delegate its remote data possession checking capability to the proxy by sending it a warrant. The warrant will be stored both in the proxy and CSS.

In this paper, we revisit the two private PDP schemes. We show that the property of correctness cannot be achieved when active adversaries are involved in these auditing systems. More specifically, an active adversary can arbitrarily tamper the cloud data and produce a valid auditing. Moreover, the malicious CSS can put the auditing right to entrust to anyone and control the delegation key and lead to the failure of the subsequent validation work. Finally, we propose a solution using two-party key agreement protocol to construct authenticated tag and resolve the weakness.

2 Related Work

Ateniese et al. defined the first Provable Data Possession (PDP) model to solve the storage problems of static files in cloud computing [1]. They divided the file into blocks, and computed a homomorphic tag [2] for each block, completed the proof of the data integrity by sampling and verifying the correspondence of the tags and blocks randomly. Juels and Kaliski proposed a provable data recovery (POR) model [3]. Instead of tagging file blocks, they inserted some sentinel blocks, and verified the integrity of the file by checking the correctness of sentinel blocks. For the sentinel blocks are one-time labels, the number of times that the file can do integrity verification is limited, related to the number of sentinel blocks. Shacham and Waters proposed an improved POR model under the security model defined, and had a very complete proof [3-4]. Erway, Kupccu, Papamanthou and Tamassia were the first to explore constructions for dynamic provable data possession [5]. This scheme is essentially a fully dynamic version of the PDP solution. Wang, Wang, Ren and Lou use the tags based on Shacham and Waters to apply the data integrity verification of dynamic files [6]. Its computation and communication were both smaller than the dynamic provable data possession scheme [5].

In 2012, Zhu, Hu, Ahn and Yu presented a cooperative PDP (CPDP) scheme based on homomorphic verifiable response and hash index hierarchy [7]. Zheng and Xu discussed how to avoid maintaining multiple copies of the same file in PDP setting [8]. Yuan and Yu showed a publicly auditable PoR with the same polynomial commitment technique [9]. Noticed too many heavy computations should be taken at the client side, Wang et al. investigated how to offload PDP schemes by securely outsourcing them to a computation server [10]. Zhang and Blanton presented a dynamic PDP scheme by employing balanced update trees [11]. Built on oblivious RAM, Cash, Kup and Wichs investigated how to ensure the latest version of the outsourced file maintained by the storage server [12]. Shi, Stefanov and Papamanthou provided a more efficient dynamic PoR based on special authenticated structures [13].

Some proposals have been proposed to address the integrity concern of the remote files in a multi-user setting. Wang, Li and Li investigated how to share data by a group of members through clouds [14]. Wang, Li and Li also proposed a secure cloud storage scheme that supports dynamic group member changes (for example, join and revocation) [15]. However, the group secret key must be delivered to all

group members, which is not desirable in practice. Wang, Li and Li presented a scheme that enables user revocation without requiring any secret information to be shared among group members [16]. Many researchers proposed other data storage auditing security models and concrete schemes [17-18]. Shen et al. presented delegable provable data possession scheme [19], in which data owner generates the delegation key for delegated verifier and store the key in CSSs for verification. Wang et al. also proposed a proxy provable data possession (PPDP) model [20] and provided a construction for it. In PPDP data owner can delegate its remote data possession checking capability to the proxy by sending it a warrant. The warrant will be stored both in the proxy and CSS.

3 Attack to the Delegable Provable Data Possession Scheme

In this section we describe the delegable provable data possession scheme, and then an attack to the scheme is present.

3.1 Scheme Description

Let G_1 and G_2 be two multiplicative cyclic groups of prime order p, g is a generator of G_1, Bilinear is a map $e: G_1 \times G_1 \to G_2$, three hash functions $H_1: \{0,1\}^* \to G_1$, $H_2: G_1 \to G_1$, $H_3: (Z_q)^* \to G_1$ The public parameter is $(e, p, q, g, G_1, G_2, H_1, H_2, H_3)$.

Key generation. The client U chooses $x \in Z_q$ as his private key sk_U, and computes g^x as his public key pk_U and $H_2(g^x)^x$ as his key token kt_U.

Tag computation. Data file F is divided into n blocks $\{m_1, m_2, \ldots, m_n\}$. Each data block m_i. is tagged to a homomorphic verifiable tag σ_i, which is identified by data identifier $d_F \in G_1$ and tag identifier $T_F \| i$, where $T_F \in \{0,1\}^*$ is tag identifier seed for *F*. The client U computes these homomorphic verifiable tags $\{\sigma_i\}$ as follows $\sigma_i = (H_1(T_F \| i) d_F^{m_i})^{sk_U}$, for $1 \le i \le n$. Then the client U uploads (*F*, d_F, $\{\sigma_i\}$) to the cloud server and holds (d_F, T_F)for identifying and verifying $\{\sigma_i\}$.

Delegation. The verifier V gives his key token kt_V to U over a secure channel, and obtains (d_F, T_F) from U. Then U computes the delegation key $dk_{U \to V} = kt_V^{1/sk_U}$ and gives it to the cloud server. The server uses pk_U and pk_V to verify validity of $dk_{U \to V}$ by checking whether $e(pk_U, dk_{U \to V}) = e(pk_V, H_2(pk_V))$. To revoke the verifier V, the client U commands the cloud server to remove $dk_{U \to V}$ from its storage directly.

Integrity Check. To check integrity of *F*, the verifier V chooses coefficients $C = (c_1, c_2, \ldots, c_n) \in Z_q^n$ and gives the cloud server the challenge *chal*=$(C, M_1, M_2) = (C, d_F^S, H_3(C)^S)$, where $s \in Z_q$. After receiving *chal*, the cloud server verifies it by checking whether $e(M_1, H_3(C)) = e(d_F, M_2)$. If so, the server uses (*F*,{ σ_i }, $dk_{U \to V}$, *chal*) to generate a proof *pf*=(ρ, E_1, E_2, E_3, E_4) = ($e(\sum_{i=1}^n \sigma_i^{c_i}, dk_{U \to V})^t$, $M_1^{\sum_{i=1}^n c_i m_i}$, $M_2^{\sum_{i=1}^n c_i m_i}$, $H_2(pk_V)^t, g^t$). And gives it to the verifier V as a response, where $t \in Z_q$. After receiving *pf*, the verifier V uses (sk_V, d_F, T_F, *C*, *s*) to verify *pf* by checking whether

$$\rho^s = e(\sum_{i=1}^n H_1(T_F \| i)^{sc_i} E_1, E_3)^{sk_V} \quad \textbf{(1)}$$

$$e(E_1, H_3(C)) = e(d_F, E_2) \quad \textbf{(2)}$$

$$e(E_3, g) = e(H_2(pk_V), E_4) \quad \textbf{(3)}$$

3.2 Attack to the Delegable PDP Scheme

A malicious cloud server and malicious verifier can conspire to produce a fake delegation key. So the malicious CSS will give a fake delegation to any user according with wishes for data integer validation. But the validation is not legitimately commissioned by data owner. Below we will describe an attack to the delegable provable data possession scheme.

The counterfeit verifier Δ generates his private key sk_Δ, public key pk_Δ and key token kt_Δ, then gives it to U, and obtains(d_F , T_F) from U. Then U computes the delegation key $dk_{U\to\Delta} = kt_\Delta^{1/sk_U}$ and gives it to the counterfeit cloud server. The server uses pk_U and pk_Δ to verify validity of $dk_{U\to\Delta}$ by checking whether $e(pk_U, dk_{U\to\Delta}) = e(pk_\Delta, H_2(pk_\Delta))$. The counterfeit cloud server gets private key sk_Δ of the counterfeit verifier Δ , and computes $dk_{U\to} = (dk_{U\to\Delta})^{1/sk_V} = H_2(pk_\Delta)^{1/sk_U}$. The server randomly selects $\zeta \in Z_q$ and computes $(dk_{U\to})^\zeta$ for a randomly verifier Γ as its delegation key $dk_{U\to\Gamma}$. Consequently Γ can verify the integrity of *F*.

Γ chooses coefficients $C' = (c_1', c_2', \ldots, c_n') \in Z_q^n$ and gives the cloud server the challenge *chal*= $(C', M_1', M_2') = (C', d_F^{s'}, H_3(C')^{s'})$, where $s' \in Z_q$. After receiving *chal*, the cloud server verifies it by checking whether $e(M_1', H_3(C')) = e(d_F, M_2')$. If so, the server uses (*F*, $\{\sigma_i\}$, $dk_{U\to\Gamma}$, *chal*) to generate a proof *pf* = $(\rho', E_1', E_2', E_3', E_4') = (e(\sum_{i=1}^n \sigma_i^{c_i'}, dk_{U\to\Gamma})^{t'}, M_1^{\sum_{i=1}^n c_i' m_i}, M_2^{\sum_{i=1}^n c_i' m_i}, H_2(pk_V)^{t'}, g^{t'})$ and gives it to the verifier Γ as a response, where $t' \in Z_q$.

After receiving *pf*, the counterfeit verifier uses (ζ , d_F , T_F , C' , s') to verify *pf* by checking whether $\rho^{s'} = e(\sum_{i=1}^n H_1(T_F \| i)^{s' c_i'} E_1', E_3')^\zeta$, $e(E_1', H_3(C')) = e(d_F, E_2')$, $e(E_3', g) = e(H_2(pk_\Delta), E_4')$.

From the above mentioned, we find that the counterfeit verifier Γ can verify the data blocks, but it did not get the legitimately commission from users.

4 Attack to the Proxy PDP Scheme

In this section we describe the proxy provable data possession scheme, and then an attack to the scheme is present.

4.1 Notation

Suppose the maximum number of the stored block-tag pairs is n. Let *f* and Ω be two pseudo-random functions, and let π be a pseudo-random permutation and *h* be a cryptographic hash function. Let *H* be a cryptographic hash function with two inputs. They can be described below.

$$f : Z_q^* \times \{1,2,\ldots,n\} \to Z_q^*,$$

$$\Omega : Z_q^* \times \{1,2,\ldots,n\} \to Z_q^*,$$

$$\pi : Z_q^* \times \{1,2,\ldots,n\} \to \{1,2,\ldots,n\},$$

$$H : G_2 \times \{0,1\}^* \to Z_q^*, h : Z_q^* \to G_1^*$$

We let $f_z(i)$ denote the function *f* with input **z** and *i*.

4.2 Scheme Description

SetUp. The client picks a random number $x \in Z_q$ and computes $X = g^x$. The client (with private/public key pair (x, X)) can delegate its remote data possession checking capability to a proxy (with private/public key pair (z, Z), $z \in Z_q$, $Z = g^z$) by sending it a warrant. The warrant consists of the description ω of the constraints. (y, Y) is the CSS's private and public key pairs. Once designated, the proxy can check the client's remote data possession.

TagGen. The client computes $t = H(e(Y,Z)^x, \omega)$, $W_i = \Omega_t(i)$, $T_{m_i} = (h(W_i)u^{m_i})^x$, the signature $Sign = Sig_x(\omega)$.

After the procedures are performed n times, all block tags are generated. Then the client sends the block-tag pairs collection $\{(m_i, T_{m_i})\}$ and the warrant ω to the CSS, where $1 \le i \le n$. The client deletes the block-tag pairs from its local storage. At the same time, the client sends the warrant-signature pair (ω, S_i) to the proxy.

SignVerify. Upon receiving the client's signature $Sign = Sig_x(\omega)$ on the warrant ω, the proxy performs the verification algorithm. If it is valid, the proxy accepts this warrant ω; otherwise, the proxy rejects it.

CheckTag. Given $\{(m_i, T_{m_i})\}$ ($1 \le i \le n$), CSS computes $t = H(e(X,Z)^y, \omega)$ and $W_i = \Omega_t(i)$. Then it verifies whether $e(T_i, g) = e(h(W_i)u^{m_i}, X)$ holds. If it holds, then CSS accepts it. Otherwise, CSS rejects it and queries the client for new block-tag pair.

GenProof. Let the challenge be $chal = (k_1, k_2, c)$, where $1 \le c \le n$, $k_1, k_2 \in Z_q^*$. The proxy sends($\omega, Sign$) to CSS. The CSS verifies whether $Sign$ is valid and compare this ω with its stored warrant. If it is false, CSS rejects the proxy's query; otherwise computes the indexes and coefficients of the blocks for which the proof is generated: $i_j = \pi_{k_1}(j)$, $a_j = f_{k_2}(j)$, $T = \sum_{j=1}^{c} T_{m_{ij}}^{a_j}$, $\hat{m} = \sum_{j=1}^{c} a_j m_{ij}$. Then CSS sends $V = (\hat{m}, T)$ to the proxy as the response to the *chal* query.

CheckProof. Upon receiving the response V from the CSS, the proxy computes $t = H(e(X,Y)^z, \omega)$. Then checks whether the following for formula holds.

$$e(T, g) = e(\sum_{i=1}^{c} h(\Omega_t(\pi_{k_1}(i)))^{f_{k_2}(i)} u^{\hat{m}}, X) \quad \textbf{(4)}$$

If it holds, then the proxy outputs "success". Otherwise the proxy outputs "failure".

4.3 Attack to the Proxy PDP Scheme

Malicious cloud server can choose (k_1, k_2, c), which has ever been used to verify data integrity. And it is sent to a fake verifier. The fake verifier performs the following validation.

(1) The malicious cloud server computes $i_j = \pi_{k_1}(j)$, $a_j = f_{k_2}(j)$, $T = \sum_{j=1}^{c} T_{m_{ij}}^{a_j}$, $\hat{m} = \sum_{j=1}^{c} a_j m_{ij}$, $t = H(e(X,Z)^y, \omega)$. Then the server outputs $V = (\hat{m}, T, t)$ and sends V to the fake verifier as the response to the fake query.

(2) Upon receiving the response V from the malicious server, the fake proxy checks whether the following formula holds:

$$e(T, g) = e(\sum_{i=1}^{c} h(\Omega_t(\pi_{k_1}(i)))^{f_{k_2}(i)} u^{\hat{m}}, X) \quad \textbf{(5)}$$

If it holds, then the fake proxy outputs "success", Otherwise "failure".

Through the above process, we make an illegal verifier to check integrity of the data blocks. But the verifier is not delegated by client.

5 Countermeasure and our scheme

In order to resolve the security problem, we take use of a Diffie-Hellman key agreement key as the foundation of the homomorphic authenticator to generate the tag for every data block, which makes that the client and the verifier can mutually check marked data blocks. In addition, the verifier is designated by client. CSS does not stores any information of the verifier, i.e. the CSS is stateless to the verifiers. Our scheme is described below.

5.1 Our Scheme

We assume that file F (potentially encoded using Reed-Solomon codes) is divided into n blocks $\{m_1, m_2, \cdots, m_n\}$, where $m_i \in Z_q$ and q is a large prime. Let G be a cyclic multiplicative group on ECC generated by g, two hash functions $H_1, H_2 : \{0,1\}^* \to Z_q$, viewed as a random oracle. The procedure of our basic scheme execution is as follows.

$KeyGen(1^k) \to (sk, pk)$

The client choose a random $x \in Z_q$ and compute $X = g^x$. The secret key is x and the public key is X. The client designates a trust verifier DV. DV run the *KeyGen* and randomly choose $y \in Z_q$ as his private key and computes $Y = g^y$ as his public key.

$TagGen(x, Y, m) \to T_m$

Given $F = \{m_1, m_2, \cdots, m_n\}$, the client generates the tag T_m of the block m_i.

Let k_{i1} and k_{i2} are random integer in Z_q. The client computes them as follows: $k_{i1} \| k_{i2} = H_1(Y^x, m_i)$. And Client compute $\sigma_{i,1} = (Y^{H_2(m_i)k_{i1}+k_{i2}})^x$ $\sigma_{i,2} = X^{k_{i1}}$, $\sigma_{i,3} = X^{k_{i2}}$, then denote the set by $\phi = \{\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3}\}$ $(1 \le i \le n)$. The client sends $T_m = \{F, \phi\}$ to the CSS and deletes them from its local storage.

$GenChal(k) \to chal$

The client or the designated verifier can verify the integrity of the outsourced data by challenging the server. Verifier picks a random subset I of the set [1, n], For $i \in I (1 \le i \le c)$, the verifier chooses a random element $v_i \in Z_q$. The verifier sends the message $chal = \{(i, v_i)\}_{i \in I}$ to the CSS.

$Genproof(F, \phi, chal) \to DV$

Upon receiving the challenge, the CSS computes $\sigma = \prod_{i=1}^{c} \sigma_{i,1}^{v_i}$, $\delta = \prod_{i=1}^{c} \sigma_{i,2}^{H_2(m_i)v_i}$, $\eta = \prod_{i=1}^{c} \sigma_{i,3}^{v_i}$.

Moreover the CSS will also provide the verifier with a small amount of metadata information. The CSS outputs $pf = \{\sigma, \delta, \eta\}$ and sends pf to the verifier as the response.

$VerifyProof(X, y, pf, chal) \to \{true, false\}$

Upon receiving the response pf from the CSS, the designated verifier checks whether the following formula holds.

$$\sigma = (\delta\eta)^y \tag{6}$$

If so, output "*true*"; otherwise "*false*".

5.2 Security Analysis

The correctness analysis and security analysis of our scheme can be given by the following theorems.

Theorem 1. If Client and CSS are honest and follow the proposed procedures, then any challenge-response can pass verifier's checking, i.e., it satisfies the correctness. If you cannot provide your figures electronically, paste originals into the manuscript and center them between the margins. For halftone figures (photos), please forward high-contrast glossy prints and mark the space in the text as well as the back of the photos clearly, so that there can be no doubt about where or which way up they should be placed.

Proof. According to our scheme procedures, we know that

$$\begin{aligned}\sigma &= \prod_{i=1}^{c} \sigma_{i,1}^{v_i} \\ &= \prod_{i=1}^{c} (Y^{(H_2(m_i)k_{i1}+k_{i2})x})^{v_i} \\ &= \prod_{i=1}^{c} (g^{(H_2(m_i)k_{i1}+k_{i2})xy})^{v_i} \\ &= \prod_{i=1}^{c} (X^{(H_2(m_i)k_{i1}+k_{i2})y})^{v_i} \\ &= \prod_{i=1}^{c} (X^{H_2(m_i)k_{i1}y})^{v_i} \prod_{i=1}^{c} (X^{k_{i2}y})^{v_i} \\ &= \prod_{i=1}^{c} (\sigma_{i,2}{}^{H_2(m_i)y})^{v_i} \prod_{i=1}^{c} (\sigma_{i,3}{}^{y})^{v_i} \\ &= (\delta\eta)^{y}\end{aligned}$$

6 Conclusion

We revisit the delegable provable data possession and the proxy provable data possession scheme. We show that the property of correctness cannot be achieved when active adversaries are involved in these auditing systems. More specifically, an active adversary can arbitrarily tamper the cloud data and produce a valid auditing. Moreover, the malicious CSS can put the auditing right to entrust to anyone and control the delegation key and lead to the failure of the subsequent validation work. We also propose a solution to resolve the weakness in these schemes. Our scheme removed expensive bilinear computing. Moreover, the verifier is stateless and independent from cloud storage server, which is an important secure property in PDP schemes.

Acknowledgement

This work is supported by the NSFC (NO. 61232016, 61300236, 61300237, 61402234), Jiangsu Province Natural Science Research Program (BK20130809, BK2012461), the prospective research projects in the future Network (BY2013095-4-04), the Project of six personnel in Jiangsu Province (2013-WLW-012), the industrial Strategic Technology Development Program (10041740) funded by the Ministry of Trade, Industry and Energy (MOTIE) Korea, the open fund project from Jiangsu Engineering Center of Network Monitoring (KJR1302, KJR1305), the research fund from Jiangsu Technology & Engineering Center of Meteorological Sensor Network in NUIST under Grant (No. KDXG1301), the PAPD fund and the national training programs of innovation and entrepreneurship for undergraduates (NO. 201510300083, 201510300060).

References

[1] G. Ateniese, R.C. Burns, R. Curtmola, J. Herring, L. Kissner, Z.N.J. Peterson, D.X. Song, Provable data possession at untrusted stores, in: Proc. ACM Conference on Computer and Communications Security, 2007.

[2] R. Johnson, D. Molnar, D. Song, D. Wagner, Homomorphic signature schemes, in: Proc. CT-RSA, 2002.

[3] A. Juels, B.S. Kaliski Jr, PORs: proofs of retrievability for large files. <http://www.arijuels.com/wp-content/uploads/2013/09/JK07.pdf>, 2007.

[4] H. Shacham, B. Waters, Compact proofs of retrievability, Journal of Cryptology 26(3)(2013) 442-483.

[5] C. Erway, A. Kupccu, C. Papamanthou, R. Tamassia, Dynamic provable data possession, in: Proc. the 16th ACM Conference on Computer and Communications Security, 2009.

[6] Q. Wang, C. Wang, J. Li, K. Ren, W. Lou, Enabling public verifiability and data dynamics for storage security in cloud computing, IEEE Transactions on Parallel and Distributed Systems 22(5)(2011) 847-859.

[7] Y. Zhu, H. Hu, G. Ahn, M. Yu, Cooperative provable data possession for integrity verification in multicloudstorage, IEEE

Transactions on Parallel and Distributed Systems 23(12)(2012) 109-127.

[8] Q. Zheng, S. Xu, Secure and efficient proof of storage with deduplication, in: Proc. the Second ACM Conference on Data and Application Security and Privacy, 2012.

[9] J. Yuan, S. Yu, Proofs of retrievability with public verifiability and constant communication cost in cloud, in: Proc. the 2013 International Workshop on Security in Cloud Computing, 2013.

[10] Y. Wang, Q. Wu, D.S. Wong, B. Qin, S.S.M. Chow, Z. Liu, X. Tan, Securely outsourcing exponentiations with single untrusted program for cloud storage, in: Proc. Computer Security–ESORICS 2014, 2014.

[11] Y. Zhang, M. Blanton, Efficient dynamic provable possession of remote data via balanced update trees, in: Proc. the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, ASIA CCS'13, 2013.

[12] D. Cash, A. Kupcu, D. Wichs, Dynamic proofs of˙ retrievability via oblivious RAM, in: Proc. Advances in Cryptology–EUROCRYPT 2013, 2013.

[13] E. Shi, E. Stefanov, C. Papamanthou, Practical dynamic proofs of retrievability, in: Proc. the 2013 ACM SIGSAC Conference on Computer and Communications Security, 2013.

[14] B. Wang, B. Li, H. Li, Knox: privacy-preserving auditing for shared data with large groups in the cloud, in: Proc. Applied Cryptography and Network Security, 2012.

[15] B. Wang, H. Li, M. Li, Privacy-preserving public auditing for shared cloud data supporting group dynamics, in: Proc. Communications (ICC), 2013 IEEE International Conference on, 2013.

[16] B. Wang, B. Li, H. Li, Public auditing for shared data with efficient user revocation in the cloud, in: Proc. INFOCOM, 2013 Proceedings IEEE, 2013.

[17] Y. Ren, J. Shen, J. Wang, J. Han, S. Lee, Mutual verifiable provable data auditing in public cloud storage, Journal of Internet Technology 16(2)(2015) 317-323.

[18] Z. Fu, X. Sun, Q. Liu, L. Zhou, J. Shu, Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing, IEICE Transactions on Communications E98-B(1)(2015) 190-200.

[19] S.T. Shen, W. Tzeng, Delegable provable data possession for remote data in the clouds, in: Proc. ICICS 2011, 2011.

[20] H. Wang, Proxy provable data possession in public clouds, IEEE Transaction of Services Computing 6(4)(2012) 551-559.