Mutual Verifiable Provable Data Auditing in Public Cloud Storage

Yongjun Ren[1,2], Jian Shen[1,2], Jin Wang[1,2], Jin Han[1,2], Sungyoung Lee[3]
[1]Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, China
[2]School of Computer and Software, Nanjing University of Information Science & Technology, China
[3]Department of Computer Engineering, Kyung Hee University, Korea
{renyj100, s_shenjian}@126.com, {wangjin, 002254}@ nuist.edu.cn, sylee@oslab.khu.ac.kr

Abstract

Cloud storage is now a hot research topic in information technology. In cloud storage, date security properties such as data confidentiality, integrity and availability become more and more important in many commercial applications. Recently, many provable data possession (PDP) schemes are proposed to protect data integrity. In some cases, it has to delegate the remote data possession checking task to some proxy. However, these PDP schemes are not secure since the proxy stores some state information in cloud storage servers. Hence, in this paper, we propose an efficient mutual verifiable provable data possession scheme, which utilizes Diffie-Hellman shared key to construct the homomorphic authenticator. In particular, the verifier in our scheme is stateless and independent of the cloud storage service. It is worth noting that the presented scheme is very efficient compared with the previous PDP schemes, since the bilinear operation is not required.

Keywords: Cloud computing, Data storage auditing, Provable data possession, Key agreement.

1 Introduction

Cloud storage is an important service of cloud computing, which allows data owners to move data from their local computing systems to the Cloud. In the cloud paradigm, data owners move the large data files from their local computing systems to the remote servers. Moreover it is of critical importance for the data owners can avoid the initial investment of expensive infrastructure setup, large equipment, and daily maintenance cost, which is particularly true for small and medium-sized businesses. In addition the data owners can rely on the Cloud to provide more reliable services, so that they can access data from anywhere and at any time [1-3].

Storing the data in cloud environment becomes natural and also essential. But, security becomes one of the major concerns for all entities in cloud services [4-6]. Firstly, data owners would worry that their data could be misused or accessed by unauthorized users. Secondly, the data owners would worry that their data could be lost in the Cloud. This is because data loss could happen in any infrastructure. Furthermore, the cloud storage service (CSS) may be dishonest and they may discard the data which has not been accessed or rarely accessed to save the storage space or keep fewer replicas than promised. As a result, data owners need to be convinced that their data are correctly stored in the Cloud. It is desirable to have data storage auditing service to assure data are correctly stored in the Cloud.

Ateniese et al. [7] defined the first Provable Data Possession model to solve the storage auditing problem of static files in cloud computing. They completed the proof of the data integrity by sampling and verifying the correspondence of the tags and blocks randomly. Juels et al. [8] also proposed a provable data recovery (POR) model. 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 [9] proposed an improved POR model under the security model defined in [8]. Erway et al. [10] were the first to explore constructions for dynamic provable data possession. This scheme is essentially a fully dynamic version of the PDP solution. It maintained a skip-list for tags, and stored the root metadata in Client's hand to prevent replay attack. Wang et al. [11] and Gu and Sheng [12] use the tags based on [9] to apply the data integrity verification of dynamic files.

According to the role of the PDP verifier, the PDP protocols are classified into two categories: private PDP and public PDP. Private PDP is necessary in some cases. For example, the data owners will be restricted to access the Internet. Recently Shen and Tzeng presented delegable provable data possession scheme [13], in which data owner generates the delegation key for delegated verifier and store the key in CSSs for verification. Huaqun Wang also proposed a proxy provable data possession (PPDP) model and provided a construction for it [14]. In PPDP, data owner can delegate its remote data possession checking capability to the proxy by sending it a warrant, which is stored in both the proxy and CSS. However, it has shown that the two private PDP schemes are insecure because the state information of the proxy or delegated verifier is controlled by a malicious CSS [15].

In this paper we propose an efficient mutual verifiable

*Corresponding author: Jin Wang; E-mail: wangjin@nuist.edu.cn
DOI: 10.6138/JIT.2015.16.2.20140918

provable data possession (MV-PDP) scheme. In the MV-PDP, verifier is stateless and independent from CSS, which solves the problem that the verifier can be optionally specified by a malicious CSS. In our scheme, the roles of data owner and verifier can be changed with each other, which enable that both of data owner and verifier can carry on certain data-centric interaction. Moreover in our design, we use ECC-based homomorphic authenticator to design PDP scheme, which does not compute expensive bilinear and consume small amount of calculation and Communications.

The rest of the paper is organized as follows. Section 2 introduces the preliminaries and Sections 3 and 4 describe our MV-PDP scheme and its security proof. Section 5 gives the performance analysis on our PPDP protocol. Finally, Section 6 concludes this paper.

2 Preliminaries

MV-PDP system model and security model are given in this section. At the same time, bilinear pairings and some corresponding difficult problems are also depicted below.

2.1 System Model

MV-PDP system consists of three different network entities: client, cloud storage server, verifier. They can be identified below.

(1) Client: an entity, which has massive data, that will be moved to cloud storage server for maintenance and computation, can be either individual consumer or organization;

(2) Cloud Storage Server: an entity, which is managed by cloud service provider, has significant storage space and computation resource to maintain the clients' data;

(3) Private Verifier (PV): an entity, which is trusted and designated to assess and expose risk of cloud storage services upon request. After verification, PV also is able to act as a client and modify the data blocks, and then re-tag them.

By hosting their data in the Cloud, clients can avoid the initial investment of expensive infrastructure setup, large equipment, and daily maintenance cost. Since the clients no longer possess their data locally, it is necessary for the clients to ensure that their data are being correctly stored and maintained. That is, clients should be equipped with certain storage auditing services so that they can periodically check the integrity of the remote data even without the existence of local copies.

In practical scenarios clients may frequently perform block-level operations on the data files. For example, in business data process, some data blocks are processed by the user U and stored in CSS. The user V verifies its integrity, further processes them. And the data blocks are re-tagged and sent to CSS by user V. Next the user U continues to verify and deal with the data blocks. In this case, it is required to switching the role between the client and PV, which will complete data-block level processing.

2.2 Security Model and Definition

The MV-PDP scheme is composed of the following algorithms.

- $KeyGen(1^k) \rightarrow (sk, pk)$
  It takes input a security parameter k and returns the corresponding private/public key pair. By running it twice, this algorithm can return the client's private/public key pair(x, X) and a PV's private/public key pair (y, Y).
- $TagGen(x, Y, m) \rightarrow (T_m)$
  This algorithm is run by the client. It takes as input its private key x, the PV's public key Y and a file block m, and outputs the tag T_m.
- $GenChal(k) \rightarrow chal$
  The PV generates a challenge to CSS for the stored data. It takes as input the public parameter k, and outputs the challenge $chal$.
- $Genproof(F, \phi, chal) \rightarrow DV$
  The CCSs generate the possession proof. It takes as inputs the public parameter, an ordered collection F of blocks and tags ϕ, and the challenge $chal$. It returns a data possession proof pf for the blocks in F that are determined by the challenge $chal$.
- $Verfyproof(X, y, pf, chal) \rightarrow \{true, false\}$
  This algorithm can be run by the PV upon receipt of the proof pf. It takes as input the client's public key X, the challenge $chal$, its own private key y and the proof pf returned from the server, and outputs "*true*" if the integrity of the file is verified as correct, or "*false*" otherwise.

After executing the above algorithms, the PV can further processes the checked data blocks and form new data blocks. Then *TagGen* algorithm is performed by PV to generate new tags. Finally the new data blocks is stored in CSS and verified by the client.This means the private verifier can mark data blocks, which belongs to the PV, then store them in CSS. Similarly, the client can verify the integrity of the data blocks.

Definition 1 (Unforgeability): A MV-PDP scheme is secure if for any (probabilistic polynomial) adversary A (i.e., malicious CCS) the probability that A wins the MV-PDP game is negligible. The MV-PDP game between the challenger C and the adversary A can be depicted as follows:

(1) SetUp: Supposing the system parameter is *params*. *KeyGen* is a private/public key pair generating algorithm. By running *KeyGen*, C can get the client's

private/public key pair (x, X), the private auditor's private/public key pair (y, Y).C keeps (x, y) confidential and sends (X, Y) to A.

(2) Queries: A adaptively makes a number of different queries to C. Each query can be one of the following.

- Hash query: A makes Hash function queries adaptively. C responds the Hash values to A;
- Proof query: A chooses challenge *chal* and obtains a valid proof with the *chal*.

(3) Challenge: C generates a challenge *chal* which defines a ordered collection. C is required to provide a possession proof for the blocks.

(4) Answer: A computes a data possession proof *pf* for the blocks indicated by *chal* and returns *pf*. In the MV-PDP game, we say that the success probability of the adversary A is negligible. i.e.,

$$Adv_A(\Pr[Verfyproof = true]) \le \varepsilon,$$

where ε is negligible.

2.3 Hardness Problem

Let G be a cyclic multiplicative group on ECC generated by P, the related complexity assumptions are as follows.

- **Computational Diffie-Hellman (CDH) Problem:** Given a randomly chosen $P \in G$, as well as aP, bP, for unknown $a, b \in Z_p$, compute abP.
- **Elliptic Curve Discrete Logarithm Problem (ECDLP):** Given points P and Q of the group in elliptic curve, find a number k such that $P^k = Q$.

3 Mutual Verifiable Provable Data Possession Scheme

Now we start to present the main idea behind our scheme. 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.

We assume that file F (potentially encoded using Reed-Solomon codes [11]) is divided into n blocks $\{m_1, ..., m_n\}$, where $m_i \in Z_p$ and q are a large prime. Let G be a cyclic multiplicative group on ECC generated by g, two hash functions H_1, H_2: $\{0,1\}^* \to Z_p$, viewed as a random oracle. The procedure of our basic scheme execution is as follows:

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

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

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

  Given $F = \{m_1, ..., m_n\}$, the client generates the tag T_m of the block m_i. Let k_{i1} and k_{i2} be random integer in Z_p. The client computes $k_{i1} \| k_{i2} = H_1(Y^x, m_i)$.

  And client computes $\sigma_{i,1} = (Y^{H_2(m_i)k_{i1}+k_{i2}})^x$, $\sigma_{i,2} = X^{k_{i1}}$, $\sigma_{i,3} = X^{k_{i2}}$, then denotes the set by $\phi = \{\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3}\}$, $1 \le i \le n$ as the tag for block m_i. 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 (DV) 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 $v_i \in Z_p$. The verifier sends the message $chal = \{(i, v_v)\}_{i \in I}$ 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.

- *Verfyproof*$(X, y, pf, chal) \to \{true, false\}$

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

$$\sigma = (\delta\eta)^y$$

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

In our construction, the Diffie-Hellman agreement key g^{xy} is the key. Only client and private verifier know its secret key (x/y) and are able to generate the tag of data blocks, or perform integrity verification. Moreover the private verifier is directly designated by clients. Thus no state information is stored in CSS.

After executing the above algorithms, the private verifier DV can further process the checked data blocks and form new data blocks. Then *TagGen* algorithm is performed by DV to generate new tags. Finally the new data blocks is stored in CSS and verified by the client.

4 Security Analyses

The correctness analysis and security analysis of our MV-PDP 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., MV-PDP satisfies the correctness.

Proof. According to our scheme procedures, we know that

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

Theorem 2. It is computational infeasible to generate a fake tag under our MV-PDP scheme.

We show that if a (t', ε')-algorithm A, operated by an adversary, can generate a fake tag under our MV-PDP scheme after making at most H_1 hash queries, at most q_T tag queries and requesting q_K setup, then there exists a (t, ε)-algorithm B that can solve the CDH problem in G with $t \le t' + q_{H_1}T_G + q_TT_G$ and $\varepsilon \ge \varepsilon'/q_Kq_{H_1}$,where one exponentiation on G takes time T_G.

Proof. Let A be a probabilistic black-box adversary who wins the tag unforgeability game with advantage ε' in time t' On input (g, g^a, g^b) the CDH algorithm B simulates A as follows:

- Setup: Given an instance (g, g^a, g^b) of the CDH problem. B sets the public parameter (G, g, q). As A requests the creation of system users, B guesses which one A will attempt a forgery against. Without loss of generality, we assume the target public key as pk_v and set it as $pk_v = g^a$. For all other public keys, we set $pk_i = g^{x_i}$ for a random $x_i \in Z_p$. Then B can invoke A to query. The total number requested is q_K.
- Query: A can query oracles O_{H_1}, O_{H_2}, O_{Tag} during his execution. B handles these oracles as follows:
- O_{H_1}: B maintains a table T_{H_1} to look up the O_{H_1} query records. B takes m_i as input, if record (*, m_i) exists, then it outputs (k_{i1}, k_{i2}). Otherwise guess if m_i is the block m^* that A will attempt to use in a forgery. If $m_i = m^*$, output $Y^{H_2(m_i)k_{i1}+k_{i2}} = g^b$; otherwise a random $y_i \in Z_p$ is selected, let $Y^{H_2(m_i)k_{i1}+k_{i2}} = g^{y_i}$, and inserts (g^{y_i}, m_i) into T_{H_1} for each $m_i \ne m^*$.
- O_{H_2}: B maintains a table T_{H_2} to look up the O_{H_2} query records. B takes $u \in (0,1)^*$ as input and outputs ρ if record (u, ρ) exists in T_{H_2}. Otherwise, B randomly selects I from Z_p and inserts (u, I) into T_{H_2}.
- O_{Tag}: B maintains a table T_{Tag} to look up the O_{Tag} query records. B takes $F = \{m_1, ..., m_n\}$ as input. For $1 \le i \le n$, if m_i has been queried to oracle O_{H_1}, B aborts. Otherwise, B randomly chooses r_1, r_2 from Z_p, and let $k_{i1} = r_1/a$, $k_{i2} = r_2/a$. So $\sigma_{i,1} = (Y^{H_2(m_i)k_{i1}+k_{i2}})^a = g^{b(H_2(m_i)r_1+r_2)}$, $\sigma_{i,2} = X^{k_{i1}}$, $\sigma_{i,3} = X^{k_{i2}}$. Then B inserts $(m_i, \sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3})$ to T_{Tag}.
- Forgery: Eventually A outputs a forgery $(pk_j, m', \sigma_{i,1}', \sigma_{i,2}', \sigma_{i,3}')$, B responses them as a challenge to A. If $v \ne j$, B guessed the wrong target user and abort. If *VerfyProof* = *false*, or $(m', \sigma_{i,1}', \sigma_{i,2}', \sigma_{i,3}')$ is the result of any O_{Tag}, B also aborts. Otherwise B let $\sigma_{i,1}' = (Y^{H_2(m_i)k_{i1}+k_{i2}})^a = (g^b)^a = g^{ab}$ be the proposed CDH problem. So $k_{i1}' \,||\, k_{i2}' = H_1(g^{ab}, m')$, $\sigma_{i,2}' = g^{ak_{i1}'}$, $\sigma_{i,3}' = g^{ak_{i2}'}$.

The probability that B will guess the target user correctly is $1/q_K$ and the probability that B will guess the forged block m' is $1/q_{H_1}$. Therefore, if A generates a fake tag with probability ε', then B solves the CDH problem with probability $\varepsilon'/q_Kq_{H_1}$. Algorithm B requires one exponentiation on G for each H_1 query, one extra exponentiation on G for each tag query, so its running time is A's running time plus $q_{H_1}T_G + q_TT_G$.

Theorem 3. For the cloud, it is computational infeasible to generate a forgery of an auditing proof under our mechanism.

If the ECDLP is (t, ε)-secure, the MV-PDP scheme is $(t - q_1t_1 - q_2t_2 - q_3t_3, 2\varepsilon)$ proof indistinguishable in the random oracle model, where (q_1, q_2, q_3) are the numbers of times that an adversary queries $(O_{H_1}, O_{H_2}, O_{proof})$-oracles, and (t_1, t_2, t_3) are the responding time to an oracle query.

Proof. Following the security model and security game defined in Subsection 2.2. Let A be probabilistic adversary to break the MV-PDP scheme. We construct an algorithm B that uses A to solve the DLP problem as follows:

- Setup: Given an instance (g, g^a) of the DLP problem. B sets the public parameter (G, g, q), data owner's key pair is $(\beta, X = g^{\alpha\beta})$ and the designated verifier's key pair is (γ, g^b), where $\beta, \gamma \in Z_p$. Then B can invoke A to query.
- Query: A can query oracles O_{H_1}, O_{H_2}, O_{proof} during his execution. B handles these oracles as follows:
- O_{H_1}: B maintains a table T_{H_1} to look up the O_{H_1} query records. B takes m_i as input, if record (*, m_i) exists, then it outputs (k_{i1}, k_{i2}). Otherwise, B randomly selects I from G and inserts (I, m_i) into T_{H_1}.
- O_{H_2}: B maintains a table T_{H_2} to look up the O_{H_2} query records. B takes $u \in (0,1)^*$ as input and outputs ρ if record (u, ρ) exists in T_{H_2}. Otherwise, B randomly selects I from Z_p and inserts (u, I) into T_{H_2}.

- O_{proof}: B maintains a table T_{proof} to look up the O_{proof} query records. B takes $chal = \{(i, v_i)\}_{i \in I}$ as input. For $1 \le i \le n$, if m_i has been queried to oracle O_{H_1}, B aborts. Otherwise, B randomly chooses r_1, r_2 from Z_p, and let $k_{i1} = r_1/\beta$, $k_{i2} = r_2/\beta$. So $\sigma_{i,1} = (X^{H_2(m_i)k_{i1}+k_{i2}})^{\gamma} = g^{a\gamma(H_2(m_i)r_1+r_2)}$, $\sigma_{i,2} = X^{k_{i1}} = g^{ar_1}$, $\sigma_{i,3} = X^{k_{i2}} = g^{ar_2}$. Then B outputs a valid proof as follows:

$$\sigma = \prod_{i=1}^{c} \sigma_{i,1}^{v_i},$$

$$\delta = \prod_{i=1}^{c} \sigma_{i,2}^{H_2(m_i)v_i},$$

$$\eta = \prod_{i=1}^{c} \sigma_{i,3}^{v_i},$$

- Challenge: After the query-1 phase, A generates a challenge $chal = \{(i^*, v_i^*)\}_{i^* \in I}$ and requests the adversary A to provide a proof.
- Answer: A returns integrity proof $pf = \{\sigma, \delta, \eta\}$ with $chal = \{(i^*, v_i^*)\}_{i^* \in I}$. Now we say if B can solve the ECDLP problem with the instance (g, g^a), i.e., get the private key a the proof pf can pass the verification procedure.

When B can achieve the private key a, b from (g, g^a) and (g, g^b), A can compute as below:

$$k_{i1}^* \| k_{i2}^* = H_1(g^{ab}, m^*),$$

$$\sigma^* = \prod_{i=1}^{i^*} (g^{ab(H_2(m_i^*)k_{i1}^* + k_{i2}^*)})^{v_i^*},$$

$$\delta^* = \prod_{i=1}^{i^*} (g^{a(H_2(m_i^*)k_{i1}^*)})^{v_i^*},$$

$$\eta^* = \prod_{i=1}^{i^*} (g^{ak_{i2}^*})^{v_i^*},$$

If B can solve the ECDLP problem, A has advantage ε' to break the MV-PDP scheme. Therefore, the reduced advantage of B is $\varepsilon = \varepsilon'/2$ and the reduced time is $t = t' + q_1t_1 + q_2t_2 + q_3t_3$.

5 Performance Evaluation

5.1 Comparison of Computation Cost

In our MV-PDP protocol, suppose there exists n message blocks. In the *TagGen* phase, the client needs to compute 3integer exponentiation and 1 multiplication. In the *GenProof* phase, the CCS needs to do 3c integer exponentiations and 3c multiplications. In the *VerifyProof* phase, the verifier needs to do 1 multiplication and 1 integer exponentiation. Other operations like hashing and permutation are omitted since they just contribute negligible computation cost. Our scheme does need to compute expensive pairing, which improves its computational efficiency considerably.

5.2 Comparison of Communication Cost

Compared to RSA, elliptic curves cryptography (ECC) has shorter key length based on the same level of security. It has been shown that 160-bit ECC provides comparable security to 1024-bit RSA. At the same time, 224-bit ECC provides comparable security to 2048-bitRSA. The communication overhead incurred mostly comes from the MV-PDP response. In MV-PDP response, the CCS needs to send 3 elements in G. The total communication is about 480 bits. This communication overhead is totally tolerable for current communication techniques. More importantly, our scheme saves the communication cost of the required delegated checking ability for data possession as in Delegable PDP and Proxy PDP. We do not consider the communication overhead incurred by storing their data on CCS. For the storing processes, the communication overheads are less than the schemes in [13-14] because our scheme does not store the delegation key or warrant. Thus, the total communication overhead of MV-PDP is more efficient.

6 Conclusion

To meet with some practiced cloud cloud storage application scenarios, we propose the mutual verifiable provable data possession scheme. First we define MV-PDP system model and security model. And then we utilize of Diffie-Hellman shared key to construct the homomorphic authenticator. In MV-PDP system, the data blocks signed by a client can be verified by a private verifier, while the data blocks signed by a verifier can also be check by a client. And the same data blocks are easy to be signed and checked by a client and the verifier in turn. Furthermore, in MV-PDP the verifier is stateless and independent of CSS. We'd like to emphasize that ECC-based homomorphic authenticator is used to design our scheme, which result in low calculation and Communications due to the fact that bilinear operation is not required.

Acknowledgements

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. N1885014041, N1885012119).

References

[1] Chia-Mu Yu, Chi-Yuan Chen and Han-Chieh Chao, *Proof of Ownership in Deduplicated Cloud Storage with Mobile Device Efficiency*, *IEEE Network Magazine*, in press.

[2] Zhang-Jie Fu, Xing-Ming Sun, Qi Liu, Lu Zhou and Jian-Gang Shu, *Achieving Efficient Cloud Search Services: Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing*, *IEICE Transactions on Communications*, Vol.E98-B, No.1, 2015, pp.190-200.

[3] Jin Wang, Yue Yin, Jian-Wei Zhang, Sungyoung Lee and R. Simon Sherratt, *Mobility Based Energy Efficient and Multi-sink Algorithms for Consumer Home Networks*, *IEEE Transactions on Consumer Electronics*, Vol.59, No.1, 2013, pp.77-84.

[4] Jian Shen, Wen-Ying Zheng, Jin Wang, Yu-Hui Zheng and Xing-Ming Sun, *An Efficient Verifiably Encrypted Signature from Weil Pairing*, *Journal of Internet Technology*, Vol.14, No.6, 2013, pp.947-952.

[5] Jin Wang, Zhong-Qi Zhang, Feng Xia, Wei-Wei Yuan and Sungyoung Lee, *An Energy Efficient Stable Election-Based Routing Algorithm for Wireless Sensor Networks*, *Sensors*, Vol.13, No.11, 2013, pp.14301-14320.

[6] Jian Li, Xiao-Long Li, Bin Yang and Xing-Ming Sun, *Segmentation-Based Image Copy-Move Forgery Detection Scheme*, *IEEE Transactions on Information Forensics and Security*, Vol.10, No.3, 2015, pp.507-518.

[7] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson and Dawn Song, *Provable Data Possession at Untrusted Stores*, *Proc. ACM CCS*, Alexandria, VA, November, 2007, pp.598-609.

[8] Ari Juels and Burton S. Kaliski, Jr., *PORs: Proofs of Retrievability for Large Files*, *Proc. ACM CCS*, Alexandria, VA, November, 2007, pp.584-597.

[9] Hovav Shacham and Brent Waters, *Compact Proofs of Retrievability*, *Journal of Cryptology*, Vol.26, No.3, 2013, pp.442-483.

[10] C. Chris Erway, Alptekin Kupcu, Charalampos Papamanthou and Roberto Tamassia, *Dynamic Provable Data Possession*, *Proc. ACM CCS*, Chicago, IL, November, 2009, pp.213-222.

[11] Qian Wang, Cong Wang, Kui Ren, Wen-Jing Lou and Jin Li, *Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing*, *IEEE Transactions on Parallel and Distributed Systems*, Vol.22, No.5, 2011, pp.847-859.

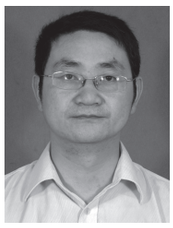
[12] Bin Gu and Victor S. Sheng, *Feasibility and Finite Convergence Analysis for Accurate On-Line v-Support Vector Learning*, *IEEE Transactions on Neural Networks and Learning Systems*, Vol.24, No.8, 2013, pp.1304-1315.

[13] Shiuan-Tzuo Shen and Wen-Guey Tzeng, *Delegable Provable Data Possession for Remote Data in the Clouds*, *Proc. of ICICS*, Beijing, China, November, 2011, pp.93-111.

[14] Huaqun Wang, *Proxy Provable Data Possession in Public Clouds*, *IEEE Transactions on Services Computing*, Vol.6, No.4, 2012, pp.551-559.

[15] Yong-Jun Ren, Jian Shen, Jin Wang and Li-Ming Fang, *Analysis of Delegable and Proxy Provable Data Possession for Cloud Storage*, *Proc. IIH-MSP*, Kitakyushu, Japan, August, 2014, pp.779-782.

Biographies



Yongjun Ren obtained the PhD degree in the computer and science department at the Nanjing University of Aeronautics and Astronautics, China, in 2008. Now he is serving as a full time faculty in the Nanjing University of Information Science and Technology. His research interests include network security and applied cryptography.

Jian Shen received the ME and PhD degrees in Computer Science from Chosun University, Korea, in 2009 and 2012, respectively. Since late 2012, he has been a professor at Nanjing University of Information Science and Technology, Nanjing, China. His research interests include computer networking, and information security systems.

Jin Wang received the MS degree from Nanjing University of Posts and Telecommunications, China in 2005. He received PhD degree from Kyung Hee University Korea in 2010. Now, he is a professor at Nanjing University of Information Science & Technology. His research interests mainly include wireless sensor network, network performance analysis and security.

Jin Han received his PhD degrees in Computer Science from Nanjing University, China in 2010. His current research focuses on Ubiquitous computing and applications, wireless Ad-hoc and sensor networks, network security and privacy, applied cryptography and security protocol analysis.

Sungyoung Lee received MS and PhD degrees in Computer Science from Illinois Institute of Technology (IIT), USA in 1987 and 1991. He has been a professor at Kyung Hee University since 1993. His current research focuses on Ubiquitous computing and applications, wireless Ad-hoc and sensor networks, and Embedded Systems.