ID Based Automated Data Integrity Checking Model

Ms. Pooja G. Natu
Department of Computer Engineering
AISSMS COE, Pune
Pune, India

Mrs. Shikha Pachouly
Department of Computer Engineering
AISSMS COE, Pune
Pune, India

Abstract- ***The cloud-based outsourced storage relieves the client's burden of storage management and integrity preservation by providing an equivalently elastic, inexpensive, location-independent platform. As clients no longer have physical control over data, outsourced data integrity checking is of vital importance in cloud storage. It allows the clients to verify data intactness and correctness with no downloading the entire data on local machine. As the verification is to be done at client end, the integrity checking protocol must be efficient to save client's time. Another aspect of the protocol is flexibility, which improves the quality of integrity checking by allowing user specific block partition size. The verifiers are not reliable and must be excluded from system. Moreover in case of company oriented scenario, maintaining log records of each verification request can help in security analysis. Taking these three points into consideration, we have proposed the flexible, automated and log based RDPC model as: ID Based ADIC as Identity Based Automated Data Integrity Checking model for single-cloud storage. The proposed model is based on bilinear pairings and RDPC technique. The approach eliminates certification management with the help of Identity management and additionally provides log management towards data integrity. The model makes client independent from initiating verification request and keeping the track of previous records which reduces client's time. The security is maintained as verifier is not involved. The principle concept here is to make data integrity checking a painless job for any client.***

Index Terms **- Remote data Possession Checking; Identity Based Management; Cloud Storage Security.**

I. INTRODUCTION

The cloud computing facilitates many straight benefits to clients as on order service, location independence, elasticity, network based model, resource pooling and so on. The cloud storage provisioning is one of the vital services of cloud computing. The cloud storage facilitates massive amount of data storage which magnetize small and middle scale organizations to utilize remote storage for efficient and economic storage management. It is a model of storage where the data is stored in logical pool, the physical storage spans multiple servers and the physical environment is actually owned and managed by a hosting entity. The tasks like keeping the data available and accessible, and the physical environment protected and running is done by cloud storage providers. Though the cloud storage is grabbing the market, many security issues hinder the client to move their data on remote server. The serious issue of data integrity comes whenever client uploads data on un-trusted servers. In such scenarios, clients need to apply strategies to prove originality of data. Also the client must not need to retrieve entire data to check integrity in order to have reduced network and computational overhead. The central trouble of remote data security is ensuring integrity of remotely located data which is out of client reach. The companies are moving their sensitive data over cloud in order to gain economic and operational benefits. Ensuring cloud users that their data is intact is especially important when users are companies. The remote data possession checking (RDPC) is a primarily designed to address the data integrity checking issue in company environment.

A. Motivation

Storing the data in cloud environment becomes natural and essential too. But, security is one of the major concerns for all units in cloud services. The data owners require worrying about misuse of data, illegal access to the data and data loss. Furthermore, the cloud service providers (CSP) may be untruthful and they may discard the data which has not been accessed or rarely accessed to save the storage space or hide considerable data loss caused during migration or for any other reason to maintain good reputation in market. As a result, data owners need to be convinced that their data are correctly stored in the Cloud. The previous studies considered many parameters but lacking in reducing client interaction and logging.

B. Related Work

In cloud computing, remote data integrity checking is an important security issue in today's taxonomy. The

client's massive data is not in local environment and outside control. The malicious cloud server may damage the client's data in order to gain more benefits and to maintain their reputation. Many researchers proposed the equivalent system model and security model to work on security issue. Latest work was proposed by Wang H. which was an ID based RDPC approach for multi cloud storage [1]. In 2007, provable data possession (PDP) paradigm was proposed by Ateniese et al. [2], where the verifier can check remote data integrity with a high probability. Based on the RSA technique, they provided two provably secure PDP schemes. As a continuation, Ateniese et al. proposed dynamic PDP model and concrete scheme [3] which failed in providing a support for insert operations. Taking this limitation in consideration, in 2009, Erway et al. proposed a full-dynamic PDP scheme based on the authenticated flip table [4]. Then F. Seb´e et al. did the similar work in [5]. The central idea of PDP is it allows a verifier to verify the remote data integrity even with no retrieving or downloading the complete data. It is based on probabilistic proof of possession by sampling random set of blocks from the server, which reduces I/O costs considerably. The verifier which may be the client needs to preserve small metadata to carry out the integrity checking tasks. The novel approach towards PDP was proposed by Wang in 2012 which was the concrete scheme of proxy PDP in public clouds [6]. As the era was moving towards multi cloud environment, Zhu et al. proposed the cooperative PDP in the multi-cloud storage which is for hybrid cloud [7]. The multiple replica PDP approach was then proposed by Ateniese et al [8].

In reference with the Ateniese et al.'s revolutionary effort, many remote data integrity checking models and protocols have been proposed [9], [10], [11], [12], [13]. Shacham offered the first proof of retrievability (POR) scheme with provable security in 2008 [14]. The idea of POR is, the verifier can check the remote data truth and even can recover the remote data at any time. The state of the art can be found in [15], [16], [17], [18]. In some scenarios, the client can delegate the remote data integrity checking job to the third party which is commonly referred as the third party auditing in cloud computing [19], [20], [21], [22].

One of the profits of cloud storage is to enable global access to data with location liberty. We have obtained the comparison on currently available PDP systems in along with rewards and drawbacks of the schemes. The approach proposed here is based on above literature survey.

C. Our Contribution

In data integrity checking tasks, the preprocessing of file, generation of challenges and verification of proof are key activities. This paper focuses on providing highly flexible security provisioning approach for company oriented surroundings where client may need to check integrity of data on timely basis. We recommend the novel RDPC model as ID Based ADIC. With the help of Identity management the protocol is ended fairly efficient. Additionally, the protocol is automated and produces and maintains log file in order to preserve verification records for an organization. The protocol provides flexibility in data preprocessing phase where user can select the size of file blocks to have projected security over the file.

D. Paper Organization

The rest of the paper is organized as follows. Section 2 formalizes the AutoID-DIC model. Section 3 presents our Auto ID-RDPC protocol with a detailed description of techniques used in the model. Section 4 presents the results of the proposed model. Finally, Section 5 concludes the paper.

II. SYSTEM STRUCTURAL DESIGN AND MODELING

A.Introduction

The Auto ID-DIC system model and the detailed description of the protocol are presented in this section. The model comprises of various entities and these entities perform required operations. The entities of system are shown in Figure 1 and can be given as:

1. Client: an entity, which has massive data to be stored on the single-cloud for maintenance and computation, in our scenario, the client is any employee of an organization. The system includes only client and verifier is not included to improve security.

2. PKG (Private Key Generator): an entity, when receiving the identity, it outputs the corresponding private key to the client. It reduces certification management overhead drastically.

3. CSS (Cloud Storage Server): an entity, which is remotely located and managed by cloud service provider. It has significant storage space and computation resource to manage the client's data. More specifically these are un-trusted and remotely located entities

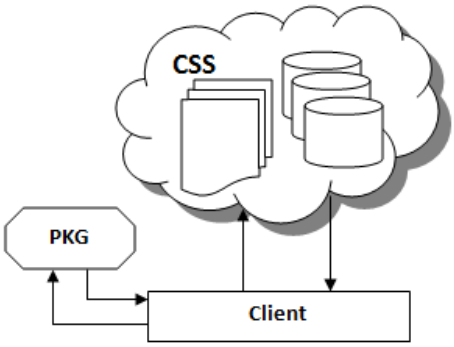

Figure 1: System Model of ID DIC

In the cloud model, clients can be relieved from the load of storage and computation by putting the big data files on the isolated cloud servers. As the clients no longer possess their data locally, it is of serious importance for them to ensure that their data is being correctly stored and will not get tainted or damaged. That is, client should be prepared with well organized faster security means to periodically verify the correctness of the distant data even with no existence of local copies. We formally define an ID Based ADIC scheme. We have then specified a security definition to absorb security requirements of the model. We have presented the environment required for the model and its implementation details.

B. Algorithm Specifications

Definition 1(ID Based ADIC): An ID Based ADIC protocol is a set of six polynomial time algorithms (Setup, Extract, FileBreak, TagGen, GenProof, CheckProof) which run as follows:

1. *(params, mpk, msk)* ← Setup(1^k) is the parameter-generation algorithm. It takes the security parameter *k* as input and gives three things: the system public parameters *params*, the master public key *mpk* and the master secret key *msk*. This action is performed by PKG in order to generate security parameters.
2. (pk_{ID}, sk_{ID}) ← Extract(1^k , *params*, *mpk*, *msk*, *ID*) is a probabilistic key-extraction algorithm that is run by PKG to extract the client's private key. It takes three inputs as: the public parameters *params*, the master public key *mpk*, the master secret key *msk*, and the identity *ID* of a client. As a consequence, Extract(.) output the private key sk_{ID} equivalent to the client with identity *ID*.
3. F ← *FileDivide(F,n)* is an algorithm that is run by client to partition the file into required number of blocks which depends on expected data security. The function takes a file and factor n as parameter and gives a set of file parts as an output. This is an improvement which makes the model extra flexible.
4. T_m ← TagGen(sk_{ID},*m*) is an algorithm that is run by the client *ID* to produce the verification metadata. It takes two input parameters: a secret key sk_{ID} and a file block *m*, and returns the verification tag T_m which is retained by client for data integrity verification.
5. *V* ← GenProof(*params, F, chal,*∑*, ID*) is run by the CSS to produce a proof of data possession. It receive as inputs the public parameter *params*, the client's identity *ID*, an prearranged collection *F* of file blocks, a challenge *chal* and an prearranged collection ∑ of the verification tag corresponding to the blocks in *F*. The function returns a proof of data possession *V* for the blocks in *F* that are determined by the challenge *chal*.
6. *{* "*success*", "*failure*"*}* ← CheckProof(*mpk, skID, chal, V*) is run by the client to validate a proof of data possession. It takes four inputs: the master public key *mpk*, the master secret key *sk*, a challenge *chal* and a proof of data possession *V* . Following verification process it returns "success" or "failure", representing that *V* is a correct proof or not. This entry is upholded in the log which is addition over standard ID RDPC approach.

There are two variations of data integrity checking policy: Public verifiability and Private verifiability. In the CheckProof, if the private key sk_{ID} is necessary, the current protocol is considered as a Private verifiability. As we are considering company oriented environment, the model we propose belongs to this type.

In addition to communication and computation overheads as low as possible, the protocol should satisfy the following requirements:

1. Less Storage at Client: The client should not be required to keep an entire copy of the file(s) to be verified on local machines. It would be unrealistic and infeasible for a client to replicate the whole content. Also, the cost will be improved. Storing a reduced-size digest of the data at the client should be adequate for verification of the CSS-generated proof.
2. Verification: The protocol has to reside protected even if the prover is malicious. A malicious prover is interested in proving awareness of some data that he does not entirely know. Here, security means that such a prover will fail in convincing the verifier on his validity.
3. It must to be possible to run the verification an limitless number of times as employees may check the data any number of times.

In the security identification, we say that the adversary A wins in the game if CheckProof(pk_{ID^*} , sk_{ID^*} , *chal*, *V*) ="*success*". And in this protocol the chances of getting result as success are insignificant.

Definition 2 ((ρ, δ) security Measure): We can say the protocol is (ρ, δ) secure if, given a fraction ρ of PCS corrupted blocks, the probability that the corrupted blocks are detected is at least δ.

III. ID ADIC PROTOCOL

The ID Based ADICDIC approach is presented in this section. The model is based on bilinear pairings, which are reviewed below.

A. Bilinear Pairing

Let $G1$ and $G2$ be two cyclic multiplicative groups with the same prime order q, *i.e.*, $|G1| = |G2| = q$.

Let $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear map [24], which satisfies the following properties:

1. Bilinearity:
   $\forall\ g_1, g_2, g_3 \in G_1$ and $a, b \in Z_q$,

   $e(g_1, g_2g_3) = e(g_2g_3, g_1) = e(g_2, g_1)e(g_3, g_1)$

   $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$

2. Non-degeneracy:
   $\exists\ g_4, g_5 \in G_1$ such that $e(g_4, g_5) \neq 1_{G2}$.

3. Computability:
   $\forall\ g_6, g_7 \in G_1$, there is an efficient algorithm to calculate $e(g_6, g_7)$.

Our ID Based ADIC scheme is constructed on the gap Diffie-Hellman group, where the computational Diffie-Hellman (CDH) difficulty is hard while the decisional Diffie-Hellman (DDH) problem is easy. A bilinear map e can be constructed with the modified Weil or Tate pairings on elliptic curves.

B. The Protocol Structure

This protocol comprises the procedures Setup, Extract, FileDivide, TagGen, SetTimer, GenProof, CheckProof and LogRecord. The protocol architecture is described as follows:

1. Initially *PKG* creates the public and private key for the client along with public parameters. In Extract phase, client sends the ID (unique identity of user) to PKG.
2. The PKG then generates the secret key sk_{ID} and sends back to client.
3. The client generates the file blocks along with corresponding block tag pairs and uploads them to PCS. Client then deletes all data from local machine and keeps only related metadata.
4. The client starts the timer for specific milliseconds.
5. The verifier who is the client in our system generates challenge and forwards the challenge to PCS on timer off event.
6. PCS then creates the appropriate possession proof for requested file block.
7. PCS sends the possession proof to the client.
8. The client on itself checks the possession proof and logs the particular entry to log file.

The architecture of proposed model is illustrated in figure 2 below:

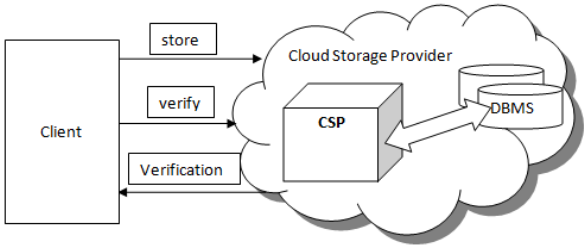

Figure 2: Architecture of our ID Based ADIC Approach

Suppose that the number of file blocks is n. We have described below the procedures of the ID Based ADIC scheme.

- Setup: PKG chooses a random number $x \in Z_q^*$ and sets $Y = g^x$, where g is a generator of the group G_1. PKG chooses a random item $u \in G_1^*$. Define two cryptographic hash functions as: $H : \{0, 1\}^* \rightarrow Z_q^*$, $h : Z_q^* \rightarrow G_1^*$. Let f be a pseudo-random function and let π be a pseudo-random permutation as given in equation (1):

  $$f : Z_q^* \times \{1, 2, \ldots, n\} \rightarrow Z_q^*$$
  $$\pi : Z_q^* \times \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, n\} \quad (1)$$

  Finally, PKG publishes
  $\{G_1, G_2, e, q, g, Y, u, H, h, f, \pi\}$ and keeps x as the *master key*.

- Extract. A client submits his identity *ID* to PKG. PKG picks $r \in Z_q^*$ and computes equation (2) as:

  $$R = g^r,\ \sigma = r + xH(ID,R) \bmod q \quad (2)$$

  PKG sends the private key $skI_D = (R, \sigma)$ to the client by a secure channel.

- FileDivide(F,n): This function uses a simple iterative algorithm to break the file in given number of blocks. The boundary of a block is fixed and must not be overloaded. The factor, that is how many blocks are to be generated is taken as n and F is actual file to be loaded on

CSP. The set F can be defined as $\{f_1,f_2,..,f_n\}$ where n is number of blocks generated from single file F.

- TagGen(sk_{ID}, *F*, *i*): For simplicity we consider that the client generates the tags successively according to the counter *i*. That is, the client generates a tag for a message block m_2 after m_1, which imply that the client maintains the latest value of the counter *i*. For m_i, the client performs the TagGen procedure as follows in equation (3):

  1) Calculate $T_i = (h(i)u^{mi})^{\sigma}$.
  (3)
  2) Output T_i and send (m_i, T_i) to the PCS.

- GenProof(*mpk*,$F = (m_1,m_2,. . . ,m_n)$, *chal*,$\sum = (T_{m1} , . . ., T_{mn})$): In this procedure, the client herself queries the PCS for a proof of data possession of *c* file blocks whose indices are randomly selected using a pseudo-random permutation keyed with a fresh randomly-chosen key for each challenge. The integer $k_1 \in Z_q^*$ is the random key of the pseudo-random permutation π. Also, $k_2 \in Z_q^*$ is the random key of the pseudo-random function *f*.
  Let *chal* = (c, k_1, k_2). Then, the PCS does:
  1) For $1 \leq j \leq c$, compute the indices and coefficients of the blocks for which the proof is generated:

  $i_j = \pi_{k1}(j)$, $a_j = f_{k2}(j)$. In this step, the challenge *chal* defines an ordered set $\{c, i_1, ..., i_c, a_1, ... , a_c\}$.

  2) Compute:

$$T = \prod_{j=1}^{c} T_{i_j}^{a_j}, \hat{m} = \sum_{j=1}^{c} a_j m_{i_j}.$$

  (4)

  3) Output $V = (T, m)$ and send *V* to the client as the response to the *chal* query.

- CheckProof(*mpk*, *skID*, *chal*, *V*): Upon receiving the response *V* from the PCS, the client does confirmation of challenge and received verification.

- Log entry is saved in log file for as file block id, time and result as success or failure.

IV. SYSTEM ANALYSIS

The model needs to get analyzed based on two major parameters: Computational cost and communication overhead. Initially, we analyze the performance of our proposed Auto ID-DPDP protocol from the computation and communication overhead. Afterwards, we examine our proposed system with previously invented systems based on properties of flexibility and verification logging.

The proposed model does not add major additional *computation* costs than standard approach. The model provides best approach in pre-processing which cause considerable computational overhead. If we have file f and client desire to break it into n blocks, the computational complexity is O(n) to divide the file. Actually, client provide value of n<c for standard file storage where c is a constant. The client computations are carried out on timer basis which added a minute complexity to deal with the timer. The approach supports logging which needs small amount of time to perform I/O operations.

The *communication* overhead caused varies which is based on amount of blocks generated by client. We have used Identity management to compact additional communication. The U.S. National Bureau of Standards and ANSI X9 have determined the shortest key length requirements: 1024 bits for RSA and DSA, 160 bits for ECC.

Simulation: To study the prototypal implementation of ID Based ADIC approach, we have simulated the protocol by using Java programming with JPair Library (JPair v1.03). In the simulation, Cloud Storage is simulated on FUJITSU Lifebook A Series Laptop with normal settings.

We have compared our approach with previous studies as listed below in table 2.

Table 2. Comparison of our scheme with other techniques

Schemes	Query	Respon-se	Verificat-ion	Stor-age	Auto-mated	Log Based
[1]	log2 n+2log2 q	$1G_1$+s $\log_2$ q	Verifier	O(n)	No	No
[2]	$3Z^*_q$ (480) +c	$1G_1$+ $1Z^*_q$(480)+c	Verifier	O(1)	No	No
Our	$3Z^*_q$ (480)	$1G_1$+ Z^*_q(480)+v	Client	O(1)	Yes	Yes

As per the standards, we analyze the communication overhead caused by our model, which mainly comes from the queries and responses. In an id Based ADIC query, the client needs to send 3 elements in Z_q* to CS. In case of response, the CS needs to respond with 1 element in G_1 and 1 element in Z_q* to the client. The total communication is about 3*160+160+320=960 bits. The scheme [2] gives considerable performance but causes additional cost c for client request initiation. If client needs to generate challenges with small durations, the constant value grows rapidly.

Our model reduces this constant time by providing timer based approach for challenge generation trigger. This amount of communication is reasonable with current communication technologies and will be fixed for any number of files uploaded on server. In Ateniese *et al.*'s scheme, it uses as many as 6*1024=6144 bits [3]. In Table 2, we compare the communication overheads of our ID Based ADIC protocol with respect to query and response time needed for the integrity checking.

V. CONCLUSION AND FUTURE WORK

This paper proposes an ID Based ADIC model appropriate for company-oriented cloud storage. We present the novel ID Based ADIC protocol proven secure under the assumption that the CDH problem is hard.

The protocol keeps the log records which provide verification analysis. The approach allows user to set time interval after which the challenge will be generated. And hence the user time is saved and data verification is performed without human intervention. The flexibility is provided by allowing user to select factor for file breaking activity. The approach focuses on file break flexibility and automation in random challenge generation.

In future, one can extend the same approach to work for multi cloud environment. Also, we can update the system to work for other data formats like audio, video and images.

REFERENCES

[1] Wang H., "Identity-Based Distributed Provable Data Possession in Multi-Cloud Storage", Services Computing, IEEE Transactions 2014, (Volume:PP, Issue 99.)

[2] Huaqun Wang, Qianhong Wu, Bo Qin , Domingo-Ferrer, J., "Identity-based remote data possession checking in public clouds", Information Security, IET (Volume:8 , Issue: 2), pp. 114 – 121, 2014.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, "Provable Data Possession at Untrusted Stores", *CCS'07*, 2007.

[4] G. Ateniese, R. DiPietro, L. V. Mancini, G. Tsudik, "Scalable and Efficient Provable Data Possession", *SecureComm 2008*.

[5] C. C. Erway, A. Kupcu, C. Papamanthou, R. Tamassia, "Dynamic Provable Data Possession", *CCS'09*, pp. 213-222, 2009.

[6] F. Seb´e, J. Domingo-Ferrer, A. Mart´ınez-Ballest´e, Y. Deswarte, J. Quisquater, "Efficient Remote Data Integrity checking in Critical Information Infrastructures", *IEEE Transactions on Knowledge and Data Engineering*, 20(8), pp. 1-6, 2008.

[7] H.Q. Wang, "Proxy Provable Data Possession in Public Clouds," *IEEE Transactions on Services Computing*, 2012.

[8] Y. Zhu, H. Hu, G.J. Ahn, M. Yu, "Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage", *IEEE Transactions on Parallel and Distributed Systems*, 23(12), pp. 2231-2244, 2012.

[9] Y. Zhu, H. Wang, Z. Hu, G. J. Ahn, H. Hu, S. S. Yau, "Efficient Provable Data Possession for Hybrid Clouds", *CCS'10*, pp. 756-758, 2010.

[10] R. Curtmola, O. Khan, R. Burns, G. Ateniese, "MR-PDP: Multiple Replica Provable Data Possession", *ICDCS'08*, pp. 411-420, 2008.

[11] A. F. Barsoum, M. A. Hasan, "Provable Possession and Replication of Data over Cloud Servers", CACR, University of Waterloo, Report2010/32, 2010. Available at http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-32.pdf.

[12] Z. Hao, N. Yu, "A Multiple-Replica Remote Data Possession Checking Protocol with Public Verifiability", *2010 Second International Symposium on Data, Privacy, and E-Commerce*, pp. 84-89, 2010.

[13] A. F. Barsoum, M. A. Hasan, "On Verifying Dynamic Multiple Data Copies over Cloud Servers", *IACR eprint report 447, 2011*. Available at http://eprint.iacr.org/ 2011/447.pdf.

[14] A. Juels, B. S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files", *CCS'07*, pp. 584-597, 2007.

[15] H. Shacham, B. Waters, "Compact Proofs of Retrievability",b*ASIACRYPT 2008*, LNCS 5350, pp. 90-107, 2008.

[16] K. D. Bowers, A. Juels, A. Oprea, "Proofs of Retrievability: Theory and Implementation", *CCSW'09*, pp. 43-54, 2009.

[17] Q. Zheng, S. Xu. Fair and Dynamic Proofs of Retrievability. *CODASPY' 11*, pp. 237-248, 2011.

[18] Y. Dodis, S. Vadhan, D. Wichs, "Proofs of Retrievability via Hardness Amplification", *TCC 2009*, LNCS 5444, pp. 109-127, 2009.

[19] Y. Zhu, H. Wang, Z. Hu, G. J. Ahn, H. Hu, "Zero-Knowledge Proofs of Retrievability", *Sci China Inf Sci*, 54(8), pp. 1608-1617, 2011.

[20] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing", *INFOCOM 2010*, IEEE, March 2010.

[21] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing", *IEEE Transactions on Parallel And Distributed Systems* , 22(5), pp. 847-859, 2011.

[22] C. Wang, Q. Wang, K. Ren, N. Cao, W. Lou, "Toward Secure and Dependable Storage Services in Cloud Computing," *IEEE Transactions on Services Computing*, 5(2), pp. 220-232, 2012.

[23] Y. Zhu, G.J. Ahn, H. Hu, S.S. Yau, H.G. An, S. Chen, "Dynamic Audit Services for Outsourced Storages in Clouds," *IEEE Transactions on Services Computing*, 2011. http://doi.ieeecomputersociety.org/10.1109/TSC.2011.51

[24] O. Goldreich, "Foundations of Cryptography: Basic Tools", Publishing House of Electronics Industry, Beijing, 2003, pp. 194-195.

[25] D. Boneh, M. Franklin, "Identity-based Encryption from the Weil Pairing", *CRYPTO 2001*, LNCS 2139, 2001, 213-229.

[26] A. Miyaji, M. Nakabayashi, S. Takano "New Explicit Conditions of Elliptic Curve Traces for FR-reduction", *IEICE Transactions Fundamentals*, pp. 1234-1243, 2001.

[27] D. Boneh, B. Lynn, H. Shacham, "Short Signatures from the Weil Pairing", *ASIACRYPT 2001*, LNCS 2248, pp. 514-532, 2001.

[28] H. W. Lim, "On the Application of Identity-based Cryptography in Grid Security", *Ph.D. dissertation*, University of London, London, U.K., 2006.

