IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

Proof of identity-based Multiple Cloud Storage Distributed Data Processing

Ms. S.Guru Pallavi Mr.N.Maneiah

Abstract— Remote data integrity checking is of crucial importance in cloud storage. It can make the clients verify whether their outsourced data is kept intact without downloading the whole data. In some application scenarios, the clients have to store their data on multi-cloud servers. At the same time, the integrity checking protocol must be efficient in order to save the verifier's cost. From the two points, we propose a novel remote data integrity checking model: ID-DPDP (identity-based distributed provable data possession) in multi-cloud storage. The formal system model and security model are given. Based on the bilinear pairings, a concrete ID-DPDP protocol is designed. The proposed ID-DPDP protocol is provably secure under the hardness assumption of the standard CDH (computational DiffieHellman) problem. In addition to the structural advantage of elimination of certificate management, our ID-DPDP protocol is also efficient and flexible. Based on the client's authorization, the proposed ID-DPDP protocol can realize private verification, delegated verification and public verification.

1 INTRODUCTION

Over the last years, cloud computing has become an important theme in the computer field. Essentially, it takes the information processing as a service, such as storage, computing. It relieves of the burden for storage management, universal data access with independent geographical locations. At the same time, it avoids of capital expenditure on hardware, software, and personnel maintenances, etc. Thus, cloud computing attracts more intention from the enterprise. The foundations of cloud computing lie in the outsourcing of computing tasks to the third party. It entails the security risks in terms of confidentiality, integrity and availability of data and service.

The issue to convince the cloud clients that their data are kept intact is especially vital since the clients do not store these data locally. Remote data integrity checking is a primitive to address this issue. For the general case, when the client stores his data on multi-cloud servers, the distributed storage and integrity checking are indispensable. On the other hand, the integrity checking protocol must be efficient in order to make it suitable for capacity-limited end devices. Thus, based on distributed computation, we will study distributed remote data integrity checking model and present the corresponding concrete protocol in multi-cloud storage.

1.1Motivation

We consider an ocean information service corporation Cor in the cloud computing environment. Cor can provide the following services: ocean measurement data, ocean environment monitoring data, hydrological data, marine biological data, GIS information, etc. Besides of the above services, Cor has also some private information and some public information, such as the corporation's advertisement. Cor will store these different ocean data on multiple cloud servers. Different cloud service providers have different reputation and charging standard. Of course, these cloud service providers need different charges according to the different security-levels. Usually, more secure and more expensive. Thus, Cor will select different cloud service providers to store its different data. For some sensitive ocean data, it will copy these data many times and store these copies on different cloud servers. For the private data, it will store them on the private cloud server. For the public advertisement data, it will store them on the cheap public cloud server. At last, Cor stores its whole data on the different cloud servers according to their importance and sensitivity. Of course, the storage selection will take account into the Cor's profits and losses. Thus, the distributed cloud storage is indispensable. In multi-cloud environment, distributed provable data possession is an important element to secure the remote data. In PKI (public key infrastructure), provable data possession protocol needs public key certificate distribution and management. It will incur considerable overheads since the verifier will check the certificate when it checks the remote data integrity. In addition to the heavy certificate verification, the system also suffers from the other

complicated certificates management such as certificates generation, delivery, revocation, renewals, etc. In cloud computing, most verifiers only have low computation capacity. Identity-based public key cryptography can eliminate the complicated certificate management. In order to increase the efficiency, identity-based provable data possession is more attractive. Thus, it will be very meaningful to study the ID-DPDP.

1.2 Related work

In cloud computing, remote data integrity checking is an important security problem. The clients' massive data is outside his control. The malicious cloud server may corrupt the clients' data in order to gain more benefits. Many researchers proposed the corresponding system model and security model. In 2007, provable data possession (PDP) paradigm was proposed by Ateniese et al.]. In the PDP model, the verifier can check remote data integrity with a high probability. Based on the RSA, they designed two provably secure PDP schemes. After that, Ateniese et al. proposed dynamic PDP model and concrete although it does not support insert operation. In order to support the insert operation, in 2009, Erway et al. proposed a full-dynamic PDP scheme based on the authenticated flip table The similar work has also been done by F. Sebe´ et al.. PDP allows a verifier to verify the remote data integrity without retrieving or downloading the whole data. It is a probabilistic proof of possession by sampling random set of blocks from the server, which drastically reduces I/O costs. The verifier only maintains small metadata to perform the integrity checking. PDP is an interesting remote data integrity checking model. In 2012, Wang proposed the security model and concrete scheme of proxy PDP in public clouds At the same time, Zhu et al. proposed the cooperative PDP in the multi-cloud storage Following Ateniese et al.'s pioneering work, many remote data integrity checking models and protocols have been proposed In 2008, Shacham presented the first proof of retrievability (POR) scheme with provable security

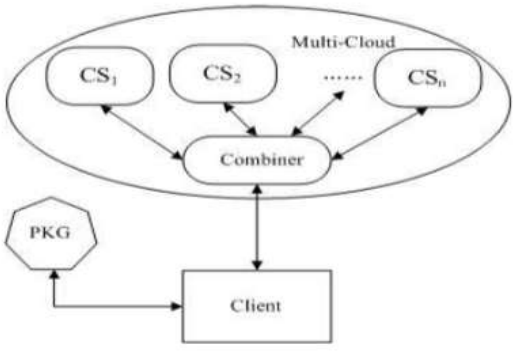

Fig. 1. The System Model of ID-DPDP

II. SYSTEM MODEL AND SECURITY MODEL OF ID-DPDP

The ID-DPDP system model and security definition are presented in this section. An ID-DPDP protocol comprises four different entities which are illustrated in Figure 1. We describe them below: 1) Client: an entity, which has massive data to be stored on the multi-cloud for maintenance and computation, can be either individual consumer or corporation. 2) CS (Cloud Server): an entity, which is managed by cloud service provider, has significant storage space and computation resource to maintain the clients' data.

3) Combiner: an entity, which receives the storage request and distributes the block-tag pairs to the corresponding cloud servers. When receiving the challenge, it splits the challenge and distributes them to the different cloud servers. When receiving the responses from the cloud servers, it combines them and sends the combined response to the verifier. 4) PKG (Private Key Generator): an entity, when receiving the identity, it outputs the corresponding private key. First, we give the definition of interactive proof system. It will be used in the definition of ID-DPDP. Then, we present the definition and security model of ID-DPDP protocol. Definition 1 (Interactive Proof System): [22] Let c,s : N → R be functions satisfying c(n) > s(n) + 1 p(n) for some polynomial p(·). An interactive pair (P,V) is called a interactive proof system for the language L, with completeness bound c(·) and soundness bound s(·), if 1) Completeness: for every x ∈ L, Pr[< P,V > (x) = 1] ≥ c(|x|). 2) Soundness: for every x 6∈ L and every interactive machine B, Pr[< B,V > (x) = 1] ≤ s(|x|). Interactive proof system is used in the definition of IDDPDP, i.e., Definition 2. Definition 2 (ID-DPDP): An ID-DPDP protocol is a collection of three algorithms (Setup, Extract, TagGen) and an interactive proof system (Proof).

4) Proof(P,C(Combiner),V (V erifier)): is a protocol among P, C and V . At the end of the interactive protocol, V outputs a bit {0|1} denoting false or true. Besides of the high efficiency based on the communication and computation overheads, a practical ID-DPDP protocol must satisfy the following security requirements:

1) The verifier can perform the ID-DPDP protocol without the local copy of the file(s) to be checked.

2) If some challenged block-tag pairs are modified or lost, the response can not pass the ID-DPDP protocol even if P and C collude. To capture the above security requirements, we define the security of an ID-DPDP protocol as follows.

C. Performance analysis

First, we analyze the performance of our proposed ID-DPDP protocol from the computation and communication overhead. We compare our ID-DPDP protocol with the other up-todate PDP protocols. On the other hand, our protocol does not suffer from resource-consuming certificate management which is require by the other existing protocols. Second, we analyze our proposed ID-DPDP protocol's properties of flexibility and verification. Third, we give the prototypal implementation of the proposed ID-DPDP protocol. Computation: Suppose there are n message blocks which will be stored in ˆ n cloud servers. The block's sector number is s. The challenged block number is c. We will consider the computation overhead in the different phases. On the group G1, bilinear pairings, exponentiation, multiplication, and the hash function h1 (the input may be large data, such as, 1G byte) contribute most computation cost. Compared with them, the hash function h, the operations on Zq and G2 are faster, the hash function H can be done once for all. Thus, we do not consider the hash functions h and H, the operations on Zq and G2. On the client, the computation cost mainly comes from the procedures of TagGen and V erification (i.e., the phase 5 in the protocol Proof(P, C, V)). In the phase TagGen, the client

III. SECURITY ANALYSIS

The security of our ID-DPDP protocol mainly consists of the correctness and unforgeability. The property of correctness has been shown in the subsection III-B. On the other hand, we study the universal unforgeability. It means that the attacker includes all the cloud servers P and the combiner C. If our ID-DPDP protocol is universally unforgeable, it can also prevent the collusion of malicious cloud servers P and C. Our proof comprises two parts: single block-tag pair is universally unforgeable; the response is universally unforgeable. Definition 7: A forger A (t,ϱ,QH,Qh,Qh1,QE,QT)breaks a single tag scheme if it runs in time at most t; A makes at most QH queries to the hash function H, at most Qh queries to the hash function h, at

most Qh1 queries to the hash function h1, at most QE queries to the extraction oracle Extract, at most QT queries to the tag oracle TagGen; and the probability that A forges a valid block-tag pair is as least ϱ. A single tag scheme is (t,ϱ,QH,Qh,Qh1,QE,QT)existentially unforgeable under an adaptive chosen-message attack if no forger (t,ϱ,QH,Qh,Qh1,QE,QT)-breaks it. The following Lemma 1 shows that the single tag scheme is secure. Lemma 1: Let (G1,G2) be a (t′,ϱ′)-GDH group pair of order q. Then the tag scheme on (G1,G2) is (t,ϱ,QH,Qh,Qh1,QE,QT)-secure against existential forgery under an adaptive chosen-block attack (in the random oracle model) for all t and ϱ satisfying ϱ′ ≥ 1 9 and t′ ≤ 23(QH+Qh)(t+O(QH)+O(QE)+O(Qh)+O(Qh1)+O(QT)) μδ(1−μ)QE (1−δ)QT ϱ , where 0 < μ,δ < 1. Based on the difficulty of the CDH problem, our single tag scheme is secure. Proof: Let the stored index-block pair set be {(1,F1), (2,F2), ⋯, (n,Fn)}. For 1 ≤ i ≤ n, each block Fi is split into s sectors, i.e., Fi = { ˜ Fi1, ˜ Fi2,⋯ , ˜ Fis}. Then, the hash function h1 is used on these sectors, i.e., Fij = h1(˜ Fij). Then, we get the hash value set {Fij}. Let the selected identity set be I = {ID1,ID2,⋯ ,ID˜ n}. We show how to construct a t′algorithm B that solves CDH problem on G1 with probability at least ϱ′. This will contradict the fact that (G1,G2) are GDH group pair. Algorithm B is given (g,ga,gb) ∈ G3 1. Its goal is to output gab ∈ G1. Algorithm B simulates the challenger and interacts with forger A as follows.

Setup. Algorithm B starts by setting the master public key as Y = ga while a keeps unknown. It initializes the three tables Tab1, Tab2 and Tab3. Then, it picks the two parameters μ and δ from the interval (0,1). H-Oracle. At any time algorithm A can query the random oracle H. To respond to these queries, algorithm B maintains a list of tuples (IDj,Rj,Hj) as explained below. We refer to this list as Tab1. When A queries the oracle H at the pair (IDj,Rj), algorithm B responds as follows: 1) If (IDj,Rj,∗) ∈ Tab1, then algorithm B retrieves the tuple (IDj,Rj,Hj) and responds with Hj, i.e., H(IDj,Rj) = Hj. 2) Otherwise, B picks a random Hj ∈ Zq. Then, it adds the tuple (IDj,Rj,Hj) to Tab1 and responds to A by setting H(IDj,Rj) = Hj. Extract-Oracle. At any time algorithm A can query the oracle Extract. To respond to these queries, algorithm B maintains a list of tuples

(ci,IDi,Ri,Hi,σi) as explained below. We refer to this list as Tab2. When A queries the oracle Extract at the identity IDi ∈ I, algorithm B picks a bit ci ∈ {0,1} according to the bivariate distribution function: Pr[ci = 0] = μ, Pr[ci = 1] = 1−μ. Based on ci, B responds as follows: 1) If ci = 1, B independently picks the random σi,Hi ∈ Zq and calculates Ri = gσi · Y −Hi. a) If (IDi,Ri,∗) 6∈ Tab1, then algorithm B adds the tuple (IDi,Ri,Hi) to Tab1 and the tuple (ci,IDi,Ri,Hi,σi) to Tab2. b) If (IDi,Ri,∗) ∈ Tab1, then algorithm B picks the other random σi,Hi ∈ Zq and calculates Ri = gσi · Y −Hi until (IDi,Ri,∗) 6∈ Tab1. Then, algorithm B adds the tuple (IDi,Ri,Hi) to Tab1 and the tuple (ci,IDi,Ri,Hi,σi) to Tab2. c) Finally, algorithm B responds with (Ri,σi). 2) If ci = 0, algorithm B independently picks a random Ri ∈ G1 and calculates Hi = H(IDi,Ri) which satisfies (IDi,Ri,∗) 6∈Tab1 (otherwise, picks another Ri and calculates H(IDi,Ri)). Then, B adds the tuple (IDi,Ri,Hi) to Tab1 and the tuple (ci,IDi,Ri,Hi,σi) to Tab2, where σi = ⊥. B fails. h-Oracle. At any time algorithm A can query the random oracle h. To respond to these queries, algorithm B maintains a list of tuples (Ni,CSli,i,zi,bi,di) as explained below. We refer to this list as Tab3. When A queries the oracle h at the tuple (Ni,CSli,i), B responds as follows: 1) If (Ni,CSli,i,zi,bi,di) ∈ Tab3 and 1 ≤ i ≤ n, then algorithm B responds with zi Qs j=1 u−Fij j , i.e., h(Ni,CSli,i) = zi Qs j=1 u−Fij j . 2) Otherwise, algorithm B picks a random bi ∈ Z∗ q and a random coin di ∈ {0,1} according to the bivariate distribution Pr[di = 0] = δ, Pr[di = 1] = 1 − δ. a) If di = 1, algorithm B calculates zi = gbi. If di = 0, algorithm B calculates zi = (gb)bi. b) Algorithm B adds the tuple (Ni,CSli,i,zi,bi,di) to Tab3 and responds with zi Qs j=1 u−Fij j , i.e., h(Ni,CSli,i) = zi Qs j=1 u−Fij j .

This completes the description of algorithm B. It remains to show that B solves the given instance of the CDH problem with a high probability. Breaking CDH: Based on the oracle-replay technique [32], B can get another block-tag pair (Fw, ˆ Tw) on the same block Fw by using different hash functions ˆ h and ˆ H. Then, we can get e(ˆ Tw,g) = e(ˆ h(Nw,CSlw,w)Qs j=1 uh1(Fwj) j ,RY H(ID,R)). According to the simulation, B knows the corresponding bw,ˆ bw that satisfy

Probability analysis. To evaluate the success probability for algorithm B, we analyze the

four events needed for B to succeed: E1: B does not abort as a result of any of A's Extract queries.

. Algorithm B's running time is the same as A's running time plus the time which is taken to respond to QH H queries, Qh h queries, Qh1 h1 queries, QE Extract queries and QT tag queries. Hence, the total running time is at most ˆ t ≤ t + O(QH) + O(QE) + O(Qh) + O(Qh1) + O(QT). By taking use of the oracle replay technique [32], A can get two different tags on the same block and randomness with the probability ϱ′ ≥ 1 9 within the time t′ ≤ 23(QH + Qh)ˆ tˆ ϱ−1, i.e., t′ ≤ 23(QH+Qh)(t+O(QH)+O(QE)+O(Qh)+O(Qh1)+O(QT)) µδ(1−µ)QE (1−δ)QT ϱ . After obtaining the two different tags on the same block and randomness, algorithm B can break the CDH problem. It contradicts the assumption that (G1,G2) is a GDH group pair. This completes the proof. Lemma 1 states that the untrusted CS has no ability to forge the single tag. But, can the untrusted CS forge the aggregated block-tag pair to cheat the verifier ? First, when all the stored block-tag pairs are challenged, we study whether the untrusted CS can aggregate the fake block-tag pairs to cheat the verifier. It corresponds to Lemma 2. Second, when some stored blocktag pairs are not challenged, we study whether the untrusted CS can substitute the valid unchallenged block-tag pairs for the modified block-tag pairs to cheat the verifier. It corresponds to Lemma 3. Lemma 2: If all the stored block-tag pairs are challenged and some block-tag pairs are modified, the malicious CS aggregates the fake block-tag pairs which are different from the challenged block-tag pairs. The combined block-tag pair (ˆ F,T) (i.e., response) only can pass the verification with negligible probability. Proof: We can use proof by contradiction to prove Lemma 2. For the challenge chal = (c,k1,k2), the following parameters can be calculated

Since the single tag is existentially unforgeable (Lemma 1), there exist at least two different indices i such that xi 6= yi. Suppose there exist ¯ s ≤ c index pairs (xi,yi) with the property xi 6= yi. Then, there exist q¯ s−1 tuples (a1,a2,··· ,ac) which satisfy the above equation (3). Since (a1,a2,··· ,ac) is a random vector, the equation (3) holds only with the probability less than q¯ s−1/qc ≤ qc−1/qc = q−1. It is negligible. Thus, if some fake block-tag pairs are aggregated, the combined blocktag

pair (ˆF,T) only can pass the verification with negligible probability. When some modified block-tag pairs are challenged and some valid block-tag pairs are unchallenged, the malicious CS still can not cheat the verifier. Lemma 3: If some challenged block-tag pairs are modified, the malicious CS substitutes the other valid block-tag pairs (which are not challenged) for the modified block-tag pairs (which are challenged). The combined block-tag pair (ˆF,T) (i.e., response) only can pass the verification with negligible probability. Proof: Define the modified block-tag pair index set as M and the valid block-tag index set as ¯ M. Let the verifier's challenge be chal = (c,k1,k2) and S = {πk1(1),⋯ ,πk1(c)}. For the CS set P, suppose some challenged block-tag pairs {(Fl,Tl),l ∈ S1 ⊆ S} are modified, i.e. S1 = STM 6= Φ. Φ denotes the empty set. The corresponding CSs (whose modified block-tag pairs are challenged) substitute the other valid block-tag pairs {(Fˆ l,Tˆ l),ˆ l ∈ S2 ⊆ ¯ M} (which are not challenged) for {(Fl,Tl),l ∈ S1} where |S1| = |S2|. By making use of the forged θi = (ˆ F(i),T(i)), the combined response θ = (ˆF,T) only can pass verification with negligible probability. If the challenged block-tag pairs (Fl,Tl),l ∈ S1 are modified, P substitutes the other valid block-tag pairs (Fˆ l,Tˆ l) for them. For 1 ≤ i ≤ c, the following parameters are calculated

ai = fk2(i), vi = πk1(i), hi = h(Nvi,CSlvi ,vi)

.

IV. CONCLUSION

In multi-cloud storage, this paper formalizes the ID-DPDP system model and security model. At the same time, we propose the first ID-DPDP protocol which is provably secure under the assumption that the CDH problem is hard. Besides of the elimination of certificate management, our ID-DPDP protocol has also flexibility and high efficiency. At the same time, the proposed ID-DPDP protocol can realize private verification, delegated verification and public verification based on the client's authorization.

REFERENCES

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, "Provable Data Possession at Untrusted Stores", CCS'07, pp. 598-609, 2007.

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

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

[4] F. Sebe´, J. Domingo-Ferrer, A. Mart´ınez-Balleste´, 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.

[5] H.Q. Wang, "Proxy Provable Data Possession in Public Clouds," IEEE Transactions on Services Computing, 2012. http://doi.ieeecomputersociety.org/10.1109/TSC.2012.35

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

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

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

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

[15] Q. Zheng, S. Xu. Fair and Dynamic Proofs of Retrievability. CODASPY'11, pp. 237-248, 2011. [16] Y. Dodis, S. Vadhan, D. Wichs, "Proofs of Retrievability via Hardness Amplification", TCC 2009, LNCS 5444, pp. 109-127, 2009.

[17] 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.

[18] 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.

Mr.N.Maneiah received M.Tech(CSE) Degree from School of Information Technology, Autonomous, and Affiliated to JNTUA, Anathapur. He is currently working as Assistant Professor in the Department of Computer Science and Engineering in Modugula Kalavathamma Institute of Technology for Women, Rajampet, Kadapa,AP. His interests includes Object Oriented Programming, Operating System, Database Management System, Computer Networking, Cloud Computing and Software Quality Assurance.

Ms. S.Guru Pallavi. She is currently pursuing M.tech Degree in Computer Science and Engineering specialization in Modugula Kalavathamma Institute of Technology for Women, Rajampet, Kadapa,AP