International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 1 | ISSN : 2456-3307

Authorized Verification of Retrieving Data in Open Cloud Environment

P. Bhavya[1], P. Chandra Prakash[2], K. Narayana[3]
[1]Student, Department of Computer Science And Engineering, Seshachala Institute Of College, Puttur, Karnataka, India
[2]Assistant Professor, Department of Computer Science And Engineering, Seshachala Inistitute Of College, Puttur, Karnataka, India

ABSTRACT

We describe a framework for knowledge and operation protection in IaaS, which includes protocols for a relied on launch of digital machines and area-based storage safety. We continue with an large theoretical evaluation with proofs about protocol resistance towards assaults in the outlined danger model. The protocols permit believe to be based by way of remotely attesting host platform configuration prior to launching guest digital machines and ensure confidentiality of knowledge in faraway storage, with encryption keys maintained external of the IaaS domain. Awarded experimental results display the validity and efficiency o0f the proposed protocols. The framework prototype was applied on a scan mattress running a public electronic health file process, displaying that the proposed protocols can also be integrated into existing cloud environments. We endorse OPoR, a new cloud storage scheme involving a cloud storage server and a cloud audit server, the place the latter is believed to be semi-sincere. In targeted, we keep in mind the assignment of permitting the cloud audit server, onbehalf of the cloud users, to pre-approach the information before uploading to the cloud storage server and later verifying the information integrity. OPoR outsources the heavy computation of the tag iteration to the cloud audit server and eliminates the involvement of user within the auditing and in the preprocessing phases. Moreover, we support the Proof of Retrievabiliy (PoR) mannequin to help dynamic information operations, as good as make sure safety towards reset assaults launched by way of the cloud storage server within the upload phase.

Keywords: Infrastructure as a Service (IaaS), Storage Provider, Retriveability

I. INTRODUCTION

Cloud computing has advanced from a bold vision to huge deployments in more than a few application domains. Nevertheless, the complexity of science underlying cloud computing introduces novel safety dangers and challenges. Threats and mitigation methods for the IaaS model were beneath intensive scrutiny in recent years [1], [2], [3], [4], at the same time the enterprise has invested in better protection solutions and issued high-quality apply strategies [5]. From an finish-person point of view the safety of cloud infrastructure implies unquestionable trust within the cloud provider, in some cases corroborated by way of stories of external auditors. At the same time vendors could present security enhancements akin to safeguard of information at relaxation, end-users have constrained or no manipulate over such mechanisms. There's a clear want for usable and cost-powerful cloud platform security mechanisms suitable for organizations that rely on cloud infrastructure. Despite the fact that

having appealing benefits as a promising provider platform for the web, this new information storage paradigm in "Cloud" brings many challenging disorders which have profound impact on the usability, reliability, scalability, security, and efficiency of the overall process. One of the crucial greatest concerns with remote knowledge storage is that of knowledge integrity verification at un trusted servers. For instance, the storage provider provider may just decide to cover such data loss incidents because the Byzantine failure from the purchasers to preserve a reputation. What is more critical is that for saving money and cupboard space the service provider might intentionally discard rarely accessed data documents which belong to an average consumer. Considering the fact that the significant size of the outsourced digital information and the patron's restricted useful resource capacity, the core of the obstacle will also be generalized as how can the patron find an effective approach to perform periodical integrity verification with out the neighborhood copy of data documents. With a view to overcome this main issue, many schemes have been proposed underneath one of a kind system and security models [1]–[10]. In all these works, high-quality efforts have been made to design solutions that meet quite a lot of specifications: excessive scheme effectivity, stateless verification, unbounded use of queries and retrievability of knowledge, and so on.

According to the position of the verifier within the mannequin, all the schemes available fall into two categories: exclusive verifiability and public verifiability. Despite the fact that reaching higher effectivity, schemes with exclusive verifiability impose computational burden on consumers. However, public verifiability alleviates purchasers from performing various computation for guaranteeing the integrity of knowledge storage. To be specific, customers are equipped to delegate a third celebration to perform the verification without devotion of their computation assets. Within the cloud, the customers could crash rapidly or can not have enough money the overload of common integrity checks. Therefore, it seems extra rational and functional to equip the verification protocol with public verifiability, which is expected to play a extra important function achieve higher efficiency for Cloud Computing. What's extra, there is another predominant predicament amongst prior designs, that's the aid of dynamic data operation for cloud data storage purposes. In Cloud Computing, the remotely stored digital data could now not only be accessed but additionally be up to date by the clients, e.G., by way of block amendment, deletion, insertion and so on. Regrettably, the-modern in the context of far flung information storage as a rule focal point on static data records and this dynamic information updates has received restrained attention in the information possession functions so far [1]–[3], [9], [11]. Though such difficulty additionally has been addressed in [12]–[14], it is good believed that supporting dynamic data operation may also be of crucial value to the functional utility of storage-outsourcing offerings. In view of the important thing function of public verifiability and dynamic information operation aid for cloud data storage, on this paper we gift a framework and an effective building for seamless integration of those two components in our protocol design. Furthermore, most of present works adopt weaker safety models which do not consider the reset assault. Particularly, the cloud storage server can trigger reset assaults within the add section to violate the soundness of the scheme.

II. PROBLEM DESCRIPTION

A. System Model

A representative network structure for cloud knowledge storage is illustrated in determine 1. Three extraordinary community entities may also be identified as follows:

purchaser: an entity that has big knowledge records to be stored within the cloud and depends on the cloud for data upkeep and computation, may also be both person purchasers or corporations.

Cloud Storage Server (CSS): an entity, which is managed by Cloud carrier supplier (CSP), has giant space for storing and computation useful resource to keep customer's data. The CSS is required to furnish integrity proof to the clients or cloud audit server for the period of the integrity checking phase.

Cloud Audit Server (CAS): a TPA, which has abilities and capabilities that customers do not need, is depended on to investigate and expose hazard of cloud storage offerings on behalf of the customers upon request. In this method, the cloud audit server also generates all the tags of the documents for the users before uploading to the cloud storage server. In the cloud paradigm, by using hanging the giant information files on the far off servers, the purchasers may also be relieved of the burden of storage and computation. As clients now not possess their information in the community, it is of significant importance for the purchasers to ensure that their knowledge are being effectively stored and maintained. That is, clients should be prepared with particular protection manner so that they may be able to periodically confirm the correctness of the far off data even with out the existence of local copies. In case that consumers don't necessarily have the time, feasibility or resources to monitor their knowledge, they may be able to delegate the monitoring mission to a trusted cloud audit server of their respective choices. In this paper, we simplest bear in mind verification schemes with public verifiability: any get together in possession of the public key can act as a verifier. We anticipate that the cloud audit server is impartial, however, the storage server is untrusted.

B. Security Definition

Shacham and Waters proposed a safety mannequin for PoR process in [3]. Mainly, the checking scheme is cozy if (i) there exists no efficient algorithm that can cheat the verifier with non-negligible likelihood; (ii) there exists a polynomial-time extractor that may recuperate the normal data file through undertaking multiple challenges responses. Under the definition of a PoR process, the client periodically challenges the storage server to make sure the correctness of the cloud information and the original records can be recovered by using interacting with the server. The definitions of correctness and soundness was once given in[3]: the scheme is right if the verification algorithm accepts when interacting with the legitimate prover (e.G., the server returns a legitimate response) and it's sound if any cheating server that convinces the consumer that is storing the data file is really storing that file.

Word that in the "game" between the adversary and the patron, the adversary has full access to the understanding stored within the server, i.E., the adversary performs the role of the prover (server). In the verification system, the goal of adversary is to cheat the purchaser, i.E., seeking to generate legitimate responses and pass the data verification without being detected. Our protection model has subtle however important change from that of the prior works. Although some earlier works additionally viewed the structure with two servers, our development achieves the outsourcing of the tag iteration. For this reason, the brand new scheme additionally requires to preclude the cloud audit server from producing invalid tags for the patron's records stored in the cloud storage server. The authentication from the cloud servers is used within the new approach to obtain this safety requirement. As a way to successfully perform the verification whilst attaining blockless, the server will have to take over the job of computing. Because of this building, our safety model differs from that of the original PoR in both the verification and the data updating process. Especially, in our scheme tags will have to be authenticated by using the consumer (prover) in each and every protocol execution as opposed to calculated or pre-stored by using the customer. Besides, our PoR model is the first to support dynamic update operations and security in opposition to reset attack in a verification scheme. The robustness against reset assault ensures that a malicious storage server can on no account attain any

knowledge of passing the verification of an incorrectly stored file with the aid of resetting the client (or the audit server) in the add segment. We will see that most of present PoR schemes can't make sure this strong safety for cloud storage.

III. THE PROPOSED SCHEME

Definition

In our scheme, both public verifiability and completely dynamic data operation are supported. We now show the definitions and parameters used in our building. (pk, sk) ← Setup(1k). It takes as enter protection parameter 1k, returns public parameters and the key pair of the cloud audit server. (F*, t) ← **upload**(sk, F). There are two phases in this algorithm. In the first section, the purchaser uploads its knowledge file F to the cloud audit server, the place F is an ordered assortment of blocks Mi. In the 2d segment, the file F is re-uploaded to the cloud storage server through the cloud audit server: it takes as enter the exclusive key sk and F, and outputs the signature set Φ, which is an ordered assortment of signatures σi on Mi. We denote the saved file F* = F,Φ. It additionally outputs metadata-the root R of a Merkle hash tree from Mi and the signature t = sigsk(h(R)) as the tag of F*. Notice that the storage server retailers (F*, t), but the audit server (the consumer) best keeps t as receipt.

1/zero ← **IntegrityVerify**P(pk, F*, t) V (pk, t). This is an interactive protocol for integrity verification of a file F* with tag t. The cloud storage server plays the function of prover P with input the public key pk, a stored file F and a file tag t. The cloud audit server performs the role of verifier V with enter pk and t. At the finish of the protocol, V outputs proper (1) if F* passes the integrity verification, or FALSE (0) otherwise.

(F∗, t) ← **update**P(pk, ˆ F*, ˆt) V (sk, ˆt, update). This is an interactive protocol for dynamic replace of a file ˆ F* with tag ˆt. The cloud storage server performs the function of prover P with enter the general public key pk, a saved file ˆ F*, and a file tag ˆt. The cloud audit server performs the function of verifier V with enter the personal key sk, ˆt, and an knowledge operation request "update" from the customer.

At the finish of the protocol, V outputs a file tag t of the up-to-date file F* if P gives a valid proof for the replace, or FALSE (0) or else. Correctness. A PoR scheme is right if the following two stipulations preserve:

• If (F*, t) ← upload(sk, F), then
IntegrityVerifyP(pk, F*, t) V (pk, t) = 1.
• If (F*, t) ← updateP(pk, ˆ F*, ˆt) V (sk, ˆt, update), then
IntegrityVerifyP(pk, F*, t) V (pk, t) = 1.

Remarks. Due to the fact the cloud audit server is fully depended on in the two-server architecture, we permit it to generate the important thing pairs on behalf of the clients within the setup segment.

Nonetheless, it possibly undesirable to position full trust on the cloud audit server in some outsourcing duties. Keep in mind the next scenario: one storage service is on hand to the customers on a pay-per-use basis, and the audit server may just upload a file, intentionally or mistakenly, on behalf of 1 customer who did not ask for storing that file. One answer for such applications is utilizing a proxy signature scheme aiding delegation by using warrant [31]– [33] to delegate the signing proper of the consumers to the cloud audit server for each utilization. The warrant to the audit server can be the hashed value of the uploaded file as a credential of the delegation.

The Core construction

Now we to gift the important suggestion behind our scheme. As within the previous PoR systems [2], [3], we count on the consumer encodes the uncooked data file e F into F using some price-ρ error correcting codes, e.G. Reed- Solomon codes. To additional minimize the computation load of the consumer, we will require that e F is pre-processed

with the aid of the cloud audit server. The encoded file F is divided into n blocks M1, . . . ,Mn, and each block has s sectors, i.E. Mi = (Mi1,Mi2, ⋆ ⋆ ⋆ ,Mis), where Mij ∈ Zp for i = 1, . . . , n, j = 1, . . . , s, and p is a large prime. Let e : G × G → GT be a bilinear map, with three cryptographic hash capabilities H, h : 0, 1∗ → G and f : zero, 1∗ → Zp, seen as random oracles [3]. Let g be the generator of G. The method of our protocol execution is as follows:

Setup: The cloud audit server chooses a random α ← Zp, u1, u2, ⋆ ⋆ ⋆ , us ← G, and computes v ← g_.
The secret secret is sk = (α) and the general public secret is pk = (v, uj1≤j≤s).
Add (phase 1: patron→ Cloud Audit Server): The customer uploads F = (M1, . . . ,Mn) to the cloud audit server. Given the file F, the cloud audit server generates a root R situated on the development of Merkle Hash Tree (MHT), the place the depart nodes of the tree are an ordered set of hashes of file blocks H(Mi) (i = 1, . . . , n). Subsequent, he signs the foundation R under his exclusive key α as h(R)_ ← sigsk(R). The file tag t = sigsk(R) is sent again to the consumer as a receipt. (section 2: Cloud Audit Server→ Cloud Storage Server): The homomorphic authenticators together
with metadata are produced as follows: for each block Mi = (Mi1,Mi2, ⋆ ⋆ ⋆ ,Mis), the cloud audit server computes a signature σi as

$$\sigma_i \leftarrow \left(H(M_i) \cdot \prod_{j=1}^{s} u_j^{M_{ij}} \right)^{\alpha} . \qquad \ldots 1$$

Denote the set of signatures by way of Φ = σi1≤i≤n. The cloud audit server sends F∗ = F,Φ to the cloud storage server. Then, the cloud audit server maintains the receipt t and deletes F∗ from its neighborhood storage. Integrity Verification: either the client or the cloud audit server can affirm the integrity of the outsourced information with the aid of challenging the cloud storage server. To generate the project query, the cloud audit server (verifier) picks

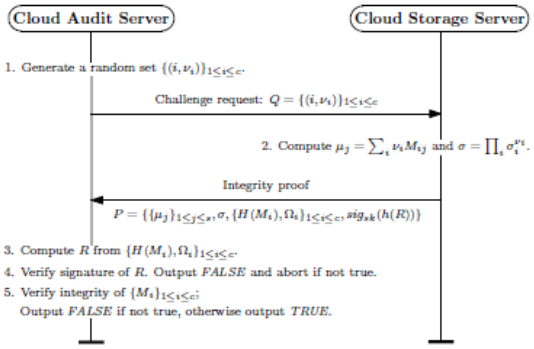

Figure 1. Protocols for Integrity Verification

A random c-element subset I of set [1, n] that denote the positions of the blocks to be checked. For each i ∈ I, picks a random element νi ← f(t, i, τ), where τ denotes the time of query. Let Q be the set (i, νi), which is shipped to the cloud storage server. Upon receiving the project question Q = (i, νi)1≤i≤c, the cloud storage server computes

$$\mu_j = \sum_{\{(i,\nu_i)\}\in Q} \nu_i M_{ij} \in \mathbb{Z}_p \qquad (2)$$

for $j = 1, \ldots, s$, and

$$\sigma = \prod_{\{(i,\nu_i)\}\in Q} \sigma_i^{\nu_i} \in G. \qquad (3)$$

In addition, the cloud storage server may also furnish the cloud audit server with a small quantity of auxiliary understanding.

The auxiliary values are the nodes siblings to the nodes H(Mi)1≤i≤c to the root R. Let Ωi1≤i≤c denote the auxiliary knowledge, the cloud storage server responds the cloud audit server with proof P = μj1≤j≤s, σ, H(Mi),Ωi1≤i≤c. Upon receiving the responses from the cloud storage server, the cloud audit server performs the following computations: (1) generates root R utilising H(Mi),Ωi1≤i≤c and checks the consistency; (2) tests if e(t, g) = e(h(R), v). (3) checks whether

$$e(\sigma, g) = e\left(\prod_{\{(i,\nu_i)\}\in Q} H(M_i)^{\nu_i} \cdot \prod_{j=1}^{s} u_j^{\mu_j}, v \right) \quad (4)$$

If the entire checking holds, output actual; or else, output FALSE. The entire protocol systems are illustrated in Figure 1.

Dynamic update: in the following, we keep in mind essentially the most common operations concerned in dynamic update, that is, data modification, knowledge insertion and data deletion.

• **data change:** believe a client intends to switch the i-th block Mi to M′
i , then the next strategies have to been carried out:
1) The client sends an update request message "update = (M, i,M′i)" to the cloud audit server, where M denotes the modification operation.
2) Upon receiving the request, the cloud audit server generates the corresponding signature
σ′

$$\sigma_i' = \left(H(M_i') \cdot \prod_{j=1}^{s} u_j^{M_{ij}'}\right)$$

, And sends replace′ = (update, σ′ i) to the storage server.
3) Upon receiving update′, the storage server
performs the next operations. He replaces the block Mi with M′ i and outputs F′. Replaces the σi with σ′ i and outputs Φ′.
Replaces H(Mi) with H(M′i) within the Merkle hash tree development and generates the brand new root R′.
– For the modification operation, replies
the client with a proof Pupdate = (Ωi,H(Mi),R′), the place Ωi is the AAI of Mi.
4) After receiving the proof Pupdate from the storage server, the cloud audit server operates as follows.
– He generates root R making use of Ωi,H(Mi).
– Authenticates R through checking if e(t, g) =
e(h(R), v).
– Computes the new root price ˆR utilizing
Ωi,H(M′i) and assessments if ˆR = R′.
– indicators the brand new root metadata R′ by means of t′ =
sigsk(R′) and sends it to the server for storage.

• **information Insertion**: believe the info proprietor needs to insert block M* after the i-th block Mi. The protocol systems are just like the data change case.

1) After receiving the proof for insert operation from the storage server, the purchaser first generates root R using Ωi,H(Mi) and authenticates R via checking if
e(t, g) = e(h(R), v).
2) If it's not genuine, output FALSE, or else the client can now assess whether or not the server has participate in the insertion as required or now not, by means of additional computing the brand new root value utilizing Ωi,H(H(Mi)‖H(M∗)) and comparing it
with R′.
3) If not, output FALSE, otherwise output authentic.
Four) The cloud auditor server indicators the brand new root metadata R′ via sigsk(R′) and sends it to the server for storage.

Knowledge Deletion: knowledge deletion is simply the reverse operation of knowledge insertion. For single block deletion, it refers to deleting the designated block and relocating all of the latter blocks one block forward. Think the server receives the update request of deleting block Mi, it'll delete Mi from its space for storing, delete the leaf node H(Mi) in the MHT and generate the brand new root metadata R′. The small print of the protocol methods are similar to these of knowledge change and insertion, that are for that reason omitted here.

IV. PERFORMANCE EVALUATION

In this part, we can provide an intensive experimental analysis of the construction proposed. We construct our testbed by using using sixty four-bit M2 high-reminiscence quadruple further huge Linux servers in Amazon EC2 platform as the auditing server and storage server, and a Linux machine with Intel(R) Core(TM)2 Duo CPU clocked at 2.40 GHz and 2 GB of system reminiscence because the consumer. So as to achieve λ = 80 bit security, the high order p of the GDH staff G of the bilinear mapping should be a hundred and sixty bits in size. Observe that in the entire reviews, the businesses G and GT are selected in 160-bit and 512-bit size respectively. Think there is a four GB file

with block measurement 4 KB, then it has n = one million blocks and s = 25 sectors every block. When it's uploaded onto the storage server, the set of signatures on the file blocks best requires for one other storage of 20 MB for information integrity verification.

For the integrity verification protocol, the question Q = (i, vi)1≤i≤c is c(lgn + 2λ) bits long. However, in the random oracle model, we are able to use a seed of twoλ bits to interchange the c-element query, and the storage server can use the hash oracle to generate the whole question after receiving the seed from the client. When λ = eighty, the query length is simplest 20 bytes. The response P = μj1≤j≤s, σ, H(Mi),Ωi1≤i≤c is 2λ(clgn+c+s+1) bits lengthy (34 KB for the four GB file). We can see that the communique rate grows just about linearly as the block measurement raises, that is commonly prompted through the increasing in size of the verification block. Within the protocol of dynamic knowledge operation, the request replace' from the audit server is 2λ(s + 1) + lgn bits (540 bytes for the 4 GB file), and the response from the storage server is Pupdate = 2λ(lgn+2) bits long (440 bytes for the 4 GB file). Furthermore, we evaluation the communique cost for each person's communication rate in Amazon EC2 cloud environment, which is 92 ms. Observe that such an overhead includes the time drinking for transmission and authentication at Amazon EC2 cloud platform. In our scan, we use ρ to denote the quite a lot of erasure code price at the same time preserving excessive detection likelihood of file corruption. In our schemes, cost ρ denotes that any ρ-fraction of the blocks suffices for file recovery. In step with [1], if t fraction of the file is corrupted, via asking proof for a regular c blocks of the file, the verifier can observe this server misbehavior with likelihood 1 – (1 – t)c. When t is set to be 1 – ρ, the chance might be (1–ρ)c. Similar to [1], 460 blocks are sufficient for the integrity verification algorithm.

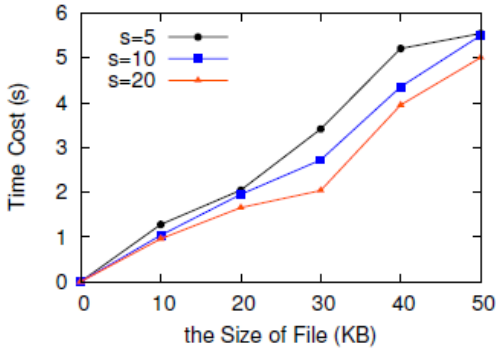

Figure 2. Tag generation time

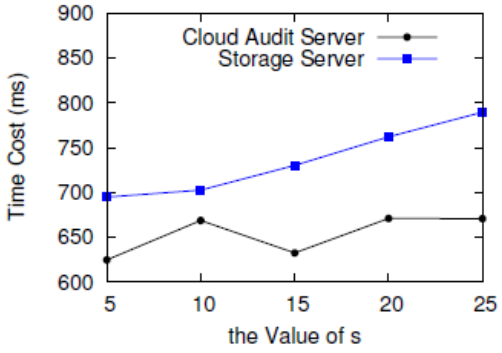

Figure 3. Verification time

Within the first scan, the computational overhead for the tag new release of records on the cloud audit server is evaluated. We now have not checked the computational overhead at users considering that it handiest wishes the computation of a digital signature, which is very small compared with the computation of the tags. The reason is that the most overhead computation has been brought to the cloud audit server. Three distinctive numbers of s are chosen in the experiment to exhibit the influence on the effectivity of the time rate. From Figure 2, we can see that the time cost grows when the number of s decreases. The normal time rate for file with dimension 50KB is 5s. When put next with the earlier associated work [12], [25], the computational overhead at customers in [12], [25] is outsourced to the cloud audit server. The response time at the user part together with the time cost of importing files, tag new release at the audit server, building of Merkle-hash tree, the conversation price and signature new release. To evaluate the response time, the cost of uploading file to the cloud audit server is

confirmed. For file with 10MB, the traditional time price is 25s. The time rate of construction for the Merklehash tree is 27s for the file with measurement 10MB. The signature generation for the basis is 3ms, which can be not noted in comparison with the time cost of importing and building of Merkle-hash tree. Be aware that the time fee of uploading records can't be avoided in any purposes.

Although the tag generation cost can also be close to the time fee of uploading documents, the cloud audit server can approach these tag new release during the file importing. As a consequence, the additional time for the response may be very small. This is appropriate for the users because the time price can be double at the consumer aspect if the tag is computed via the users. We further overview the efficiency for verification at each cloud audit server and cloud storage server in a scalable procedure in Fig.. Definitely, because the progress of the quantity of s in system, the time price for response worth at cloud storage server is increasing. That is when you consider that it wishes to compute all the exponentiations for every block in a tag. Whereas, such cost at cloud audit server is practically constant (close to 650 ms) on account that 460 blocks are enough for the integrity verification it doesn't matter what is the size of file to be checked.

V. CONCLUSION

This paper proposes OPoR, a brand new proof of retrievability for cloud storage, in which a trustworthy audit server is offered to preprocess and add the info on behalf of the purchasers. In OPoR, the computation overhead for tag iteration on the consumer part is diminished tremendously. The cloud audit server also performs the information integrity verification or updating the outsourced information upon the purchasers' request. Apart from, we construct one more new PoR scheme demonstrated cozy under a PoR mannequin with greater security in opposition to reset assault in the upload phase. The scheme also supports public verifiability and dynamic data operation simultaneously. There are a number of interesting subject matters to do alongside this study line. For illustration, we are able to (1) scale back the trust on the cloud audit server for more everyday applications, (2) support the safety model against reset attacks within the data integrity verification protocol, and (three) in finding extra effective constructions requiring for much less storage and conversation price. We depart the learn of these issues as our future work.

VI. REFERENCES

[1]. M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M.Zaharia, "A View of Cloud Computing," Comm. ACM, vol. 53, no. 4, pp. 50-58, Apr. 2010.

[2]. S. Kamara and K. Lauter, "Cryptographic Cloud Storage," Proc. Int'l Conf. Financial Cryptography and Data Security (FC), pp. 136-149, Jan. 2010.

[3]. S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing," Proc. IEEE INFOCOM, pp. 534-542, 2010.

[4]. M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable Secure File Sharing on Untrusted Storage," Proc. USENIX Conf. File and Storage Technologies, pp. 29-42, 2003.

[5]. E. Goh, H. Shacham, N. Modadugu, and D. Boneh, "Sirius: Securing Remote Untrusted Storage," Proc. Network and Distributed Systems Security Symp. (NDSS), pp. 131-145, 2003.

[6]. G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," Proc. Network and Distributed Systems Security Symp. (NDSS), pp. 29-43, 2005.

[7]. R. Lu, X. Lin, X. Liang, and X. Shen, "Secure Provenance: The Essential of Bread and Butter

of Data Forensics in Cloud Computing," Proc. ACM Symp. Information, Computer and Comm. Security, pp. 282-292, 2010.

[8]. B. Waters, "Cipher text-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization," Proc. Int'l Conf. Practice and Theory in Public Key Cryptography Conf. Public Key Cryptography, http://eprint.iacr.org/2008/290.pdf, 2008.

[9]. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," Proc. ACM Conf. Computer and Comm. Security (CCS), pp. 89-98, 2006.

[10]. D. Naor, M. Naor, and J.B. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers," Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO), pp. 41-62, 2001.

[11]. D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," Proc. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO), pp. 213-229, 2001.

[12]. D. Boneh, X. Boyen, and H. Shacham, "Short Group Signature," Proc. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO), pp. 41-55, 2004.

[13]. D. Boneh, X. Boyen, and E. Goh, "Hierarchical Identity Based Encryption with Constant Size Cipher text," Proc. Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT),pp. 440-456, 2005.

[14]. C. Delerablee, P. Paillier, and D. Pointcheval, "Fully Collusion Secure Dynamic Broadcast Encryption with Constant-Size Ciphertexts or Decryption Keys," Proc. First Int'l Conf. Pairing-Based Cryptography, pp. 39-59, 2007.

[15]. D. Chaum and E. van Heyst, "Group Signatures," Proc. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT), pp. 257-265, 1991.

[16]. A. Fiat and M. Naor, "Broadcast Encryption," Proc. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO), pp. 480-491, 1993.

[17]. B. Wang, B. Li, and H. Li, "Knox: Privacy-Preserving Auditing for Shared Data with Large Groups in the Cloud," Proc. 10th Int'l Conf. Applied Cryptography and Network Security, pp. 507-525, 2012.

[18]. C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," Proc. IEEE INFOCOM, pp. 525-533, 2010.

[19]. B. Sheng and Q. Li, "Verifiable Privacy-Preserving Range Query in Two-Tiered Sensor Networks," Proc. IEEE INFOCOM, pp. 46- 50, 2008.

[20]. D. Boneh, B. Lynn, and H. Shacham, "Short Signature from the Weil Pairing," Proc. Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology, pp. 514-532, 2001.

[21]. D. Pointcheval and J. Stern, "Security Arguments for Digital Signatures and Blind Signatures," J. Cryptology, vol. 13, no. 3, pp. 361-396, 2000.

[22]. The GNU Multiple Precision Arithmetic Library (GMP), http:// gmplib.org/, 2013.

[23]. Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL), http://certivox.com/, 2013.

[24]. The Pairing-Based Cryptography Library (PBC), http://crypto. stanford.edu/pbc/howto.html, 2013.