Secure attribute based Data sharing between dynamic groups through untrusted sources

v.spandhana
M.tech in Software Engineering
Aurora's Technological & Research Institute,
parvathapur, uppal, Hyderabad-500039

B.Malathi
Sr. Asst.professorin CSE DEPARTMENT
Aurora's Technological & Research Institute,
parvathapur, uppal, Hyderabad-500039

Abstract: Several trends are opening up the era of cloud computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the Software as a Service (SaaS) computing architecture, are transforming data centers into pools of computing service on a huge scale. The increasing network bandwidth and reliable yet flexible network connections make it even possible that users can now subscribe high quality services from data and software that reside solely on remote data centers. In order to achieve the assurances of cloud data integrity and availability and enforce the quality of cloud storage service, we consider the problem of building a secure cloud storage service on top of a public cloud infrastructure where the service provider is not completely trusted by the customer. We describe, at a high level contract signing protocol that combine recent and non-standard cryptographic primitives in order to achieve our goal.

Index Terms—Data integrity, dependable distributed storage, error localization, data dynamics, cloud computing.

I. INTRODUCTION

Several trends area unit gap up the time of Cloud Computing, that is associate degree Internet-based development and use of engineering. The ever cheaper and more powerful processors, beside the software system as a service (SaaS) computing design, area unit remodelling datacenters into pools of computing service on a large scale. The increasing network information measure and reliable nevertheless flexible network connections build it even potential that users will currently subscribe top quality services from knowledge and software system that reside exclusively on remote knowledge centres. Moving knowledge into the cloud offers nice convenience to users since they don't need to care regarding the complexities of direct hardware management. The pioneers of Cloud Computing vendors, Amazon straightforward Storage Service and Amazon Elastic reckon Cloud area unit each documented examples. Whereas these internet-based on-line services do offer vast amounts of space for storing and customizable computing resources, this computing platform shift, however, is eliminating the responsibility of native machines for knowledge maintenance at a similar time. As a result, users' area unit at the mercy of their cloud service suppliers for the supply and integrity of their knowledge.

On the one hand, although the cloud infrastructures area unit rather more powerful and reliable than personal computing devices, broad vary of both internal and external threats for data integrity still exist samples of outages and data loss incidents of noteworthy cloud storage services appear from time to time. On the opposite hand, since users could not retain a neighborhood copy of outsourced data, there exist various incentives for cloud service providers (CSP) to behave unfaithfully towards the cloud users concerning the standing of their outsourced data. For instance, to increase the profit margin by reducing cost, it's possible for CSP to discard rarely accessed data without being detected in a timely fashion.

Similarly, CSP could even attempt to hide data loss incidents thus as to maintain a reputation. Therefore, although outsourcing data in to the cloud is economically enticing for the cost and complexity of long-run large-scale data storage, its lacking of providing sturdy assurance of data integrity and convenience could impede its wide adoption by both enterprise and individual cloud users. In order to attain the assurances of cloud data integrity and convenience and enforce the quality of cloud storage service, economical ways that enable on demand data correctness verification on behalf of cloud users have to be designed. However, the fact that users no longer have physical possession of data within the cloud prohibits the direct adoption of traditional science primitives for the aim of data integrity protection. Hence, the verification of cloud storage correctness must be conducted without specific information of the full data files. Meanwhile, cloud storage is not just a third party data warehouse. The information keep within the cloud may not only be accessed but also be of updated by the users, as well as insertion, deletion, modification, appending, etc. Thus, it's additionally imperative to support the integration of this dynamic feature into the cloud storage correctness assurance that makes the system style even more difficult.

Last however not the least; the preparation of Cloud Computing is steam-powered by information centres running in a very coincidental, cooperated and distributed manner. It's additional blessings for individual users to store their information redundantly across multiple physical servers therefore on cut back the information integrity and handiness threats. Thus, distributed protocols for storage correctness assurance are going to be of most importance in achieving strong and secure cloud storage systems. However, such vital space remains to be fully explored within the literature. Recently, the importance of making certain the remote information integrity has been highlighted by the subsequent analysis works beneath totally different system and security

models. These techniques, whereas is helpful to make sure the storage correctness while not having users possessing local data, square measure all specializing in single server situation. They may be helpful for quality-of-service testing, but doesn't guarantee the information handiness just in case of server failures. Though direct applying these techniques to distributed storage(multiple servers) can be straightforward, the resulted storage verification overhead would be linear to the amount of servers.

As an complementary approach, researchers have conjointly planned distributed protocols for making certain storage correctness across multiple servers or peers. However, while providing economical cross server storage verification and information handiness insurance, these schemes square measure all specializing in static or deposit information. As a result, their capabilities of handling dynamic information remain unclear, which inevitably limits their full relevance in cloud storage situations. In this paper, we have a tendency to propose an efficient and versatile distributed storage verification theme with specific dynamic data support to make sure the correctness and handiness of users' information within the cloud. we have a tendency to consider erasure correcting code within the file distribution preparation to supply redundancies and guarantee the information dependableness against Byzantine servers , wherever a storage server may fail in absolute ways that. This construction drastically reduces the communication and storage overheads compared to the normal replication-based file distribution techniques. By utilizing the homomorphism token with distributed verification of erasure-coded information, our scheme achieves the storage correctness insurance as well as information error localization: whenever information corruption has been detected throughout the storage correctness verification, our theme will nearly guarantee the coincidental localization of knowledge errors, i.e., the identification of the misbehaving server(s).

So as to strike an honest balance between error resilience and information dynamics, we further explore the pure mathematics property of our token computation and erasure-coded information, and demonstrate the way to efficiently support dynamic operation on information blocks, while maintaining an equivalent level of storage correctness assurance. So as to save lots of the time, computation resources, and even the connected on-line burden of users, we conjointly offer the extension of the planned main scheme to support third-party auditing, wherever users will safely delegate the integrity checking tasks to TPA and be free to use the services. Our work is among the primary few ones during this field to contemplate distributed information storage security in Cloud Computing.

Our contribution is summarized as the following 3 aspects:

1) Compared to several of its predecessors, which only provide binary results regarding the storage standing across the distributed servers, the planned theme achieves the integration of storage correctness insurance and information error localization, i.e., the identification of misbehaving server(s).

2) In contrast to most previous works for making certain remote information integrity, the new theme any supports secure and efficient dynamic operations on information blocks, including update, delete and append.

3) The experiment results demonstrate the planned scheme is very economical. In depth security analysis shows our theme is resilient against Byzantine failure, malicious information modification attack, and even server colluding attacks.

II. RELATED WORK

Juels and Kaliski Jr. [10] described a formal "proof of retrievability" (POR) model for ensuring the remote data integrity. Their scheme combines spot-checking and error correcting code to ensure both possession and retrievability of files on archive service systems. Shacham and Waters[17] built on this model and constructed a random linear function-based homomorphism authenticator which enables unlimited number of challenges and requires less communication overhead due to its usage of relatively small size of BLS signature. Bowers et al. [18] proposed an improved framework for POR protocols that generalizes both Juelsand Shacham's work. Later in their succeeding work, Bowers et al. [23] extended POR model to distributed systems. However, all these schemes are focusing on static data. The effectiveness of their schemes rests primarily on the preprocessing steps that the user conducts before outsourcing the data file F. Any change to the contents of F, even few bits, must propagate through the error correcting code and the corresponding random shuffling process, thus introducing significant computation and communication complexity. Recently, Dodis et al. [20] gave theoretical studies on generalized framework for different variants of existing POR work. Ateniese et al. [11] defined the "provable data possession"(PDP) model for ensuring possession of file on untrusted storages. Their scheme utilized public key-based homomorphic tags for auditing the data file.

However, the pre computation of the tags imposes heavy computation overhead that can be expensive for an entire file. In their subsequent work, Ateniese et al. [14] described a PDP scheme that uses only symmetric key-based cryptography. This method has lower overhead than their previous scheme and allows for block updates, deletions, and appends to the stored file, which has also been supported in our work. However, their scheme focuses on single server scenario and does not provide data availability guarantee against server failures, leaving both the distributed scenario and data error recovery issue unexplored. The explicit support of data dynamics has further been studied in the two recent work [15] and [16]. Wang et al. [15]proposed to combine BLS-based homomorphic authenticator with Merkle Hash Tree to support fully data dynamics, while Erway et al. [16] developed a skip list-based scheme to enable provable data possession with fully dynamics support. The incremental cryptography work done by Bellare et al. [36] also provides a set of cryptographic building blocks such as hash, MAC, and signature functions that may be employed for storage integrity verification while supporting dynamic operations on data.

However, this branch of work falls into the traditional data integrity protection mechanism, where local copy of data has to be maintained for the verification. It is not yet clear how the work can be adapted to cloud storage scenario where usersno longer have the data at local sites but still need to ensure the storage correctness efficiently in the cloud.

The Storage and Computation Cost of Token Pre computation for 1 GB Data File under Different System Settings In other related work, Curtmola et al. [19] aimed to ensure data possession of multiple replicas across the distributed storage system. They extended the PDP scheme to cover multiple replicas without encoding each replica separately, providing guarantee that multiple copies of data are actually maintained. Lillibridge et al. [25] presented aP2P backup scheme in which blocks of a data file are dispersed across m þ k peers using an erasure code. Peers can request random blocks from their backup peers and verify the integrity using separate keyed cryptographic hashes attached on each block. Their scheme can detect data loss from free-riding peers, but does not ensure all data are unchanged. Filho and Barreto [37] proposed to verify data integrity using RSA-based hash to demonstrate unchea table data possession in peer-to-peer file sharing networks.

However, their proposal requires exponentiation over the entire data file, which is clearly impractical for the server whenever the file is large. Shah et al. [12], [13] proposed allowing a TPA to keep online storage honest by first encrypting the data then sending a number of pre computed symmetric-keyed hashes over the encrypted data to the auditor. However, their scheme only works for encrypted files, and auditors must maintain long-term state. Schwarzand Miller [24] proposed to ensure static file integrity across multiple distributed servers, using erasure-coding and block-level file integrity checks. We adopted some ideas of their distributed storage verification protocol. However, our scheme further support data dynamics and explicitly studies the problem of misbehaving server identification, while theirs did not. Very recently, Wang et al. [31] gave a study on many existing solutions on remote data integrity checking, and discussed their pros and cons under different design scenarios of secure cloud storage services. Portions of the work presented in this paper have previously appeared as an extended abstract in [1]. We have revised the paper a lot and add more technical details as compared to [1]. The primary improvements are as follows: First, we provide the protocol extension for privacy preserving third-party auditing, and discuss the application scenarios for cloud storage service. Second, we add correctness analysis of proposed storage verification design. Third, we completely redo all the experiments in our performance evaluation part, which achieves significantly improved result as compared to [1]. We also add detailed discussion on the strength of our bounded usage for protocol verifications and its comparison with state of the art.

III. Exploration of the proposed Contract Signing Protocol

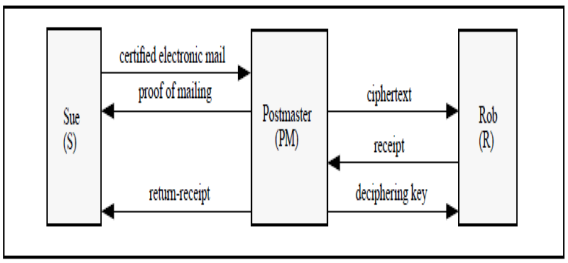
A fair contract signing protocol allows two potentially mistrusted parities to exchange their commitments. Contract signing is actually simple due to the survival of "simultaneity". That is, both parties usually sign two hard copies of the same indenture at the same place and at the same time based on the RSA signature scheme, a new digital contract signing protocol.

Here the fair exchange, among two (or multiple) potentially mistrusted parities exchanging digital items more than computer networks in a fair way, so that each revelry gets the other's item, or neither party does. In the fair exchange will contain:

1) Contract Signing Protocol.
2) Certified e-mail systems.
3) Non-reputation Protocol.

I Certified e-mail systems:

Certified electronic mail enables two equally suspicious users to exchange a acceptance for electronic mail. One family of protocols, the believers' protocols, use a faith third party. The second family, the skeptics' protocols, uses no third party. Our protocols are secure in a very strong sense; the probability of one gathering cheating can be made arbitrarily small. The protocols offer a practical example of the use of various inventive cryptographic techniques, including digital signatures, bit assurance, and zero-knowledge interactive proofs. These protocols can be executed in modern communication networks.



II Non-reputation Protocol:

The goal of a non-repudiation check is to collect, uphold, make available, and validate irrefutable evidence concerning the transfer of a message from the originator to the recipient, perhaps involving the check of a trusted third party called the Delivery Agent. We discriminate between the following non-repudiation services.

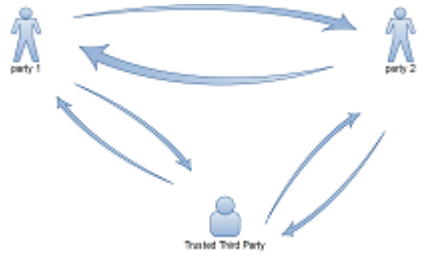
III Contract Signing Protocol:

Contract signing protocol is essentially implied by fair exchange of digital signatures between two potentially mistrusted parities with the Trusted Third Party (TTP).Two Parties a and b want to sign a contract c over a communication network. They must simultaneously exchange their commitments to c. since simultaneous exchange is usually impossible, protocols are needed to approximate simultaneity by exchanging partial commitments in piece by piece manner. During such a protocol, one party or another may have a slight

advantages a fair protocol keeps this advantage within acceptable limits

Contract signing Protocol was divided into:

1) Gradual Exchange without any TTP.
2) Protocol with an online TTP.
3) Protocol with off line TTP.



1) Gradual Exchange Without any TTP:

Gradual Exchange protocols to meet computational fairness: Both parties exchange their commitments/secrets "bit-by-bit". If one party stops prematurely, both parties have about the same fraction of the peer's secret, which means that they can complete the contract off-line by investing about the same amount of computing work. The major advantage of this approach is that no TTP is involved. However, this approach is unrealistic for most real-world applications due to the following several reasons. First of all, it is assumed that the two parties have equivalent computation resources. Otherwise, such a protocol is favorable to the party with stronger computing power, who may conditionally force the other party to commit the contract by its own interest. At the same time, such protocols are inefficient because the costs of computation and communication are extensive.

2) Protocol with an online TTP:

An on-line TTP is always involved in every exchange. In this scenario, a TTP is essentially a mediator.

- Each party first sends his/her item to the TTP
- The TTP checks the validity of those items
- If all expected items are correctly received, the TTP finally forwards each item to the party

3) Protocol with off line TTP:

This protocol is optimistic in the sense that the TTP is not invoked in the execution of exchange unless one of the two parties misbehaves or the communication channel is out of order. Trusted Third Party (TTP) is not invoked when the two involved parties perform the protocol correctly. This kind of protocol is more practical than those in which TTP mediates all transactions.

In the above said protocols are very difficult to manage. So new contract signing protocol for two mutually distrusted parties. Our protocol is based on an RSA multisignature, which is formally proved to be secured and optimistic because.

- Fairness
- Optimism
- Abuse freeness
- Provable Security
- Timely Termination
- Compatibility
- TTP'Statelessness
- High Performance

To exchange the signatures we use:

1) Registration protocol
2) Signature exchange protocol
3) Dispute resolution protocol

1) **Registration Protocol**: At first parties should Register at TTP and get certificate from the TTP. Registration protocol is a little complicated, we remark that this stage needs to be executed only once for a sufficiently long period.
2) **Signature exchange protocol:** the contract explicitly contains the following information: a predetermined but reasonable deadline , the identities of parties and the TTP. Our signature exchange protocol is briefly illuminated in Figure 1.

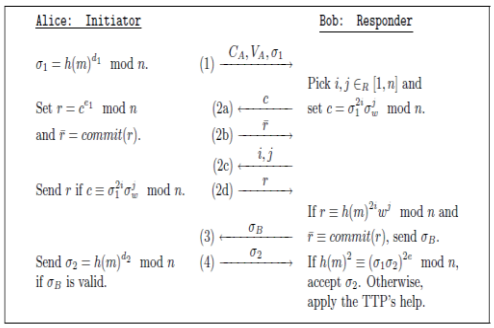


Figure 1: Signature Exchange Protocol.

3) **Dispute resolution protocol:** If party has sent his signature another party but does not receive the value before the deadline, then he sends the TTP to apply dispute resolution. Upon receiving application, the TTP performs.
   1) TTP first verifies.
   2) The TTP checks whether the deadline expires or not

3) If expires Get valid from the TTP directly by initiating dispute resolution protocol.
4) Run the Signature Exchange Protocol Again.
5) Exchange the Signatures.

Contract signing between Client and Cloud: a signcryption approach:

The entire process starts here with the employment of RSA signature algorithm [42] otherwise known as Signcyption. Here, the 1st user splits his private key d into d1 and d2 such that d=d1+d2 by following park[40]. The signature of this user has to be exchanged with the other and this signature is $\sigma_A = h(m)^{d1} \bmod n$. The partial signature generated by the 1st user is to assure that he has zero-knowledge base and this is done by Gennaro protocol[27].The connections we have are unreliable due to network failure or router's attacks [36],[46]. But, TTP is reliable since the messages inserted reach the destination for sure but with some delay.

A. Registration Protocol:

The receiver of the information has only to register i.e. only the registration of the initiator with TTP is enough. He then gets a long-term voucher along with CA. After this, the following processes are done: (for our convenience, let the sender be CLOUD and receiver as CLIENT.)

i. Client first sets an RSA modulus $n = pq$, where *p* and *q* are two -bit safe primes, i.e., there exist two primes p' and q' such that $p = 2p'+1$, $q = 2q'+1$. After, Client selects her random public key $e \in_R \square^*_{\phi(n)}$, and calculates her private key $d = e^{-1} \bmod \phi(n)$, where $\phi(n) = (p-1)*(q-1)$. At last, Client registers her public key with a CA to get her certificate C_A, which binds her identity and the corresponding pubic key (n,e) together.

ii. Client randomly splits d into $d1$ and $d2$ such that $d = d1 + d2 \bmod \phi(n)$ by choosing $d1 \in_R \square^*_{\phi(n)}$, and computes $e_1 = d_1^{-1} \bmod \phi(n)$. She also generates a sample message-signature pair (ω, σ_ω), where $\omega \in \square^*_n \setminus \{1,-1\}, ord(\omega) \geq p'q'$ and $\sigma_\omega = \omega^{d1} mod\, n$. Then, Client sends $(C_A, \omega, \sigma_\omega, d2)$ to the TTP but keeps (d, d_1, d_2, e_1) secret.

iii. The TTP first checks for the validation of Client's certificate C_A. After that, the TTP checks that the triple $(\omega, \sigma_\omega, d2)$ s prepared correctly. If everything is in correct order as per its rules, TTP saves d2 and generates a voucher V_A by computing $V_A = Sign_{TTP}(C_A, \omega, \sigma_\omega)$. This proves the TTP's signature on message $(C_A, \omega, \sigma_\omega)$, which guarantees that the TTP can issue a valid partial signature on behalf of Client by using the secret $d2$.

B. Signature Exchange Protocol:

Before all this, a contract has to be agreed between Cloud and Client and they should sign it. It should also has a deadline, and identify the Client, Cloud, and TTP.

a) Initially, the initiator Client has to compute her partial signature $\sigma_1 = h(m)^{d1} \bmod n$, and then sends the triple $(C_A, \omega, \sigma_\omega)$ to the responder Cloud. Here, $h(.)$ is a cryptographically secure hash function.

b) After receiving (C_A, V_A, σ_1), Cloud first verifies that C_A is whether issued by CA, and V_A is Client's voucher created by the TTP. Then, Cloud checks if the identities of Client, Cloud, and the TTP are correctly mentioned as part of the contract '*m'*. If all these checking are ok, Cloud initiates the below interactive zero-knowledge protocol with Client to check whether σ_1 is Client's valid partial signature on contact.

i) Then Cloud selects two numbers $i, j \in_R [1,n]$ at random, and a challenge c to Client is sent by computing $c = \sigma_1^{2i} \sigma_w^{\ j} \bmod n$.

ii) Receiving the challenge c, Client calculates the response $r = c^e \bmod n$ She then returns her commitment $\bar{r} = TCcom(r,t)$ to Cloud using a random number t, where $TCcom$ is the commitment algorithm.

iii) After receiving the commitment $\bar{r}$, Cloud sends Client the pair (i, j) to acknowledge that he is done with the challenge c properly.

iv) Client verifies for correct preparation of c, that is $c \equiv \sigma_1^{2i} \sigma_\omega^j \bmod n$. If ok, Client withdraws his commitment $\bar{r}$ by knowing the responses (r,t) to Cloud. With this (r,t), Cloud knows σ_1 as valid if and only if $r \equiv h(m)^{2i} \omega^j \bmod n$ and $\bar{r} \equiv TCcom(r,t)$.

c). Cloud checks the σ_1 Client's valid partial signature and the deadline t mentioned in contract m is whether enough for resolving the dispute resolution from the TTP. Then only he sends his signature σ_B to Client.

d). After receiving σ_B, Client has to check whether it is Cloud's valid signature. If it is, she sends Cloud the partial signature σ_2 by computing $\sigma_2 = h(m)^{d2} \bmod n$. As Cloud receives σ_2, he sets $\overline{\sigma_A} = \sigma_1 \sigma_2 \bmod n$, and accepts σ_2 as valid if and only if $h(m)^2 = \overline{\sigma_A}^{2e} \bmod n$. Here, Cloud can receive Client's standard RSA signature σ_A on message m from $\overline{\sigma_A}$. If all this do not happen, Cloud seeks the help of TTP for connection before the expiry of the date.

IV. Results and Discussion:

CASE 1:CLIENT IS HONEST, BUT CLOUD IS CHEATING.

If Cloud cheats in any possible way, he cannot learn other information except μ is valid Upon receiving the valid value of M_1, Cloud has to make a choice whether he should send his signature M_B on contract m to Client. If Cloud does, honest nitiator Client returns back her second partial signature $M_2=h(m)^{d2}$ as Cloud expects. In such a situation, Cloud gets Client's signature on contract m by setting $\mu_A = \mu_1 \mu_2$ mod n while Client also obtains Cloud's signature μ_B simultaneously. If Cloud does not send ^B or only sends an incorrect | μ_B to Client, he cannot get the value of μ_2 from ice. Furthermore, in this setting, Cloud also cannot get the value of M_2 from the TTP so that Client does not obtain his signature | μ_B. Once those values are submitted, Cloud indeed gets μ_2 from the TTP but Client receives (m, | μ_B) from the TTP, too. Therefore, once again, Cloud and Client get the other's signature on contract m at the same time

CASE 2: CLOUD IS HONEST, BUT CLIENT IS CHEATING.

In our signature exchange protocol, Client may cheat in any or some of the following steps: step (i), step (2), and step (4). First of all, according to the specification of our signature exchange protocol, to get the signature on contract from the honest responder Cloud, the initiator Client has to convince Cloud accepting as a valid partial signature in step (2). Step (2) is confirmation protocol for RSA undeniable signatures,and that their protocol satisfies the property of soundness. The soundness means that the possible cheating Client (prover), even computationally unbounded, cannot convince Cloud (verifier) to accept an invalid as valid with nonneg- ligible probability. Therefore, we conclude that to get from Cloud, Client has to send valid ^1 (with valid CAandVA) instep (1) and perform honestly in step (2). [1] Cong Wang, Qian WangKui Ren, Ning Cao, and Wenjing Lou "Toward Secure and Dependable Storage Services in Cloud Computing" IEEE transactions on services computing, vol. 5, no. 2, april-june 2012

Client is not so silly by preparing and sending ^1 to Cloud. Cloud can drive her private key (and then compute signature | μ_B. Therefore, to get signature μB from Cloud, Client has to compute and send it to Cloud. In this situation, Cloud receives valid from Client before Client gets valid μB from Cloud. After that, step (4) is the only one possi cheating chance for Client, i.e., she may refuse to reve or just send an incorrect μB to Cloud. However, this behavior does not harm Cloud essentially, since he can get the value of μB from the TTP via our dispute resolution protocol. The reason is that Cloud has received valid ^1before he sends μB to Client. After getting the value of from the TTP, Cloud can recover Client's

V. Conclusion

In this paper, we investigated the problem of data security in cloud data storage and data transmission, which is essentially a distributed storage system. To ensure the correctness of users' data in cloud data storage, we proposed an effective and flexible distributed scheme. Our scheme achieves the integration of storage correctness insurance and data error localization. In the data transmission proposed, method transferred data is encrypted in the upper-layer on top of the transport layer instead of using IPSec or SSL. Thus, the scheme for the performance improvement can be applied without modifying the implementation of IP layer, and efficient secure communications by pre-processing of encryption in the upper-layer are realized. We have used file uploading as service as web application, the security is applied over to the data at the background using the encryption algorithms like AES, Triple DES and DES. Through detailed security and performance analysis, we show that our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks. We believe that data storage security in Cloud Computing, an area full of challenges and of paramount importance, is still in its infancy now, and many research problems are yet to be identified. Adding secure cloud storage using the proposed cryptographic solution and with a searchable encryption technique for the files to be accessed, it will work as a better approach to the user to ensure security of data. The cloud security using cryptography is already in use for secure data storage which can be enhanced for secure data transmission and storage. An interesting question in this model is if we can construct a scheme to achieve both public verifiability and storage correctness assurance of dynamic data. Besides, along with our research on dynamic cloud data storage, we also plan to investigate the problem of fine-grained data error localization.

References

[2] Qian Wang,Cong Wang, Kui Ren, Wenjing Lou Jin Li "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing" IEEE transactions on parallel and distributed systems, vol. 22, no. 5, may 2011

[3] Boris Tomas1and Bojan Vuksic2 "Peer to Peer Distributed Storage and Computing Cloud System" International conference on information technology interfaces, june 25-28, 2012, cavtat, Croatia

[4] "Security and Privacy Challenges in Cloud Computing Environments" co-published by the IEEE computer and reliability ieee november/december 2010

[5] Subashini S, Kavitha V., "A survey on security issues in service delivery models of cloud computing," Journal of Network and Computer Applications (2011) vol. 34 Issue 1, January 2011 pp. 1-11.

[6] Balachander R.K, Ramakrishna P, A. Rakshit, "Cloud Security Issues, IEEE International Conference on Services Computing (2010)," pp. 517-520.

[7] Kresimir Popovic, Željko Hocenski, "Cloud computing security issues and challenges," MIPRO 2010, pp. 344-349.

[8] Amazon.com, "Amazon Web Services (AWS)," Online at http://aws. amazon.com, 2008.

[9] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres1, Maik Lindner, "A Break in Clouds: Towards a cloud Definition," ACM SIGCOMM Computer Communication Review, vol. 39, Number 1, January 2009, pp. 50-55.

[10] Patrick McDaniel, Sean W. Smith, "Outlook: Cloudy with a chance of security challenges and improvements," IEEE Computer and reliability societies (2010), pp. 77-80.

[11] Sameera Abdulrahman Almulla, Chan Yeob Yeun, "Cloud Computing Security Management," Engineering systems management and its applications (2010), pp. 1-7.

[12] Steve Mansfield-Devine, "Danger in Clouds", Network Security (2008), 12, pp. 9-11.

[13] Anthony T. Velte, Toby J.Velte, Robert Elsenpeter, Cloud Computing: A Practical Approach, Tata Mc GrawHill 2010.

[14] Gary Anthes, "Security in the cloud," In ACM Communications (2010), vol.53, Issue11, pp. 16-18.

[15] Lombardi F, Di Pietro R. Secure virtualization for cloud computing. Journal of Network Computer Applications (2010), doi:10.1016/j.jnca.2010.06.008.

