Volume 4, Issue 8, August 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Securing Privacy by Using DSA Algorithms for Cloud Storage

Sai Divya Manasa. T
Department of Computer Science and Engineering
SVECW
Bhimavaram, India

Dr. Murthy Y. S. S. R
Department of Computer Science and Engineering
SVECW
Bhimavaram, India

Abstract----Cloud computing is cyberspace based cloud computing that which enables sharing of services has many users place their data in to the cloud. But, the fact is that users can no longer have possession of the large size of deployable data that makes the data integrity protection in cloud computing a very challenging and potentially alarming task, especially for the users with limited computing resources and capabilities. So security and correctness of data are major concern. Security in cloud is achieved by signing the data block before sending to the cloud. By using the Storage in cloud, users can remotely store their data and enjoy the on-demand high quality applications and services from a shared pool of configurable resources, without the charge of local data storage and maintenance. However, the fact that users will no longer have possession of the data which makes the data integrity in cloud that is a major task, mainly for users with limited computing resources. Moreover, clients or customers should be able to use the cloud storage as if it is local or origin, without worrying about the need to integrity verification. Thus, providing public auditability for cloud storage is of major importance. So that users can resort to a third party auditor (TPA) to check the integrity of outsourced data and be worry-free. In this paper, we propose a secure cloud storage system supporting privacy-preserving public by using DSA algorithms. Here, we use the DSA algorithms to give the data which is secured, imposter prevention and authentication.

Keywords---- cyberspace, public audibility, cloud computing, third party auditor, privacy preserving

I. INTRODUCTION

Cloud computing is a next generation technology in IT industry where the present companies are been very much keen into changing the systems to the cloud due to its advantages for both the organization and also to the client or user like, resource elasticity, user usage based pricing, risk management. It is also changing the present process of saving every time to the only once to be permanently stored. One of the main goals of transferring the data is to be centralized to the cloud. From the view of the client or the developers, the storing of data is being flexible in the cloud which brings the advantages like:

1. Relief to the storage management
2. Universal data access without the dependence of geographical places
3. Reduction of capital expenditure on personnel, hardware etc.,

While cloud makes the above advantages it also brings the challenges when talking about security. This is because; cloud providers of service are different entities. As users will no longer possess the storage of the data, security cannot be directly adopted [1]. It is sometimes insufficient to search for the data errors while accessing the data. The large size of data and the resource capability of users can be expensive. The overhead of using storage in the cloud to be minimized as it reduces the user to perform many actions or operations. For the easy purpose or managing the cloud server can provide the verification request from a single designated division which is a desirable way.

In order to ensure the resources of the cloud user and also the integrity of the data, t is very important to have third party auditor (TPA) who have the capabilities and also can expertise the process where the user cannot check the integrity of the data, checks the data and integrity on behalf of users or clients. This provides an easier way for the users to believe the correctness of the storage in the cloud. The TPA would also be beneficial for the cloud service providers as it improve their platform and also to settle the conflicts with other if any. In addition to user there exists a third party Auditor to check the correctness of the process or data. For the protection of the data the users can rely on the TPA for the storage security and to avoid the unauthorized information leakage [2]. But using this cannot completely solve the problem about the privacy protecting. So in order to reduce that, we would like to say about the privacy preserving and also the data encryption in this paper.

In order to address the problem the public key will be send to the user’s mail id and when the user get’s logged in his/her homepage he need to enter the number in order to avoid the unauthorized entries in to the cloud. Mainly we would like to provide the privacy preserving auditing protocol and also to achieve the batch auditing thirdly we provide the security. We use the DSA algorithm in this paper to provide security, authentication.

II. PROBLEM STATEMENT

2.1system model

Generally, for this system model, there exist three important entities. They are user, server and the TPA (third party auditor). The user is the one who has large amount of data to store in the cloud. Server is managed by the administrator of the organization or a service in order to provide storage space and also the resources. The third party auditor is trusted because of his capabilities and expertise will access the cloud storage service. The users are reliable on cloud server, where the interaction to access the stored data for various applications. In order to save the online burden and resources, the users are resorted to TPA for the integrity and also to the preserve the privacy of data.

For the own benefits of the process, the server neglects to keep the rarely accessed data files which belong to the ordinary cloud users. The TPA is used in the business of auditing, is independent and has no incentive to collude with the server during the process of auditing. For the authorization of the server to the audit delegation to the TPA, the user need to sign the form to allow the rights to the TPA's public key where the audits are authenticated for the TPA.

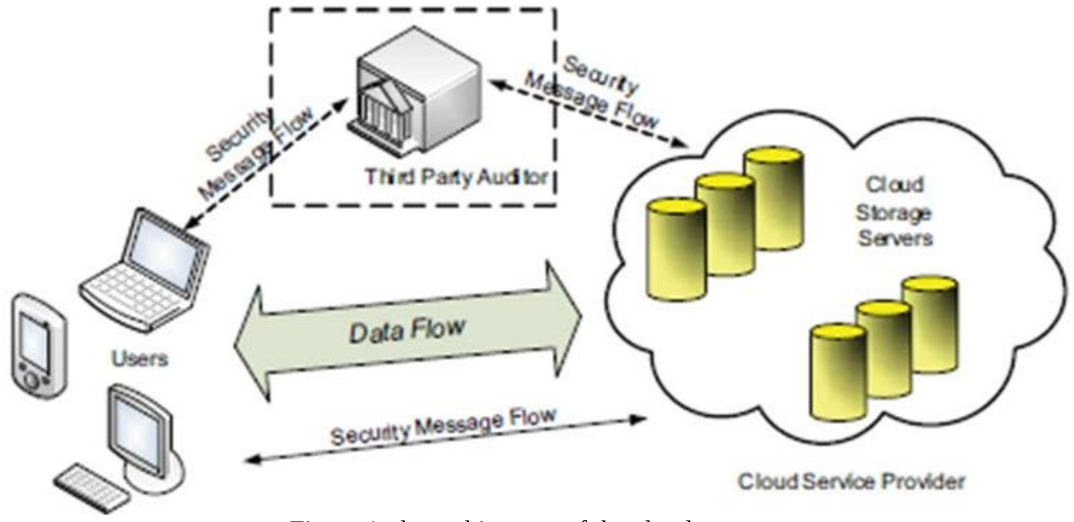

Figure 1: the architecture of the cloud storage

2.2 design goals

The following are the design goals for the protocol. They are:

1) To allowing in verify the data by TPA for the correctness without having to retrieve the entire copy of data and also without online burden additionally.
2) To ensure that the cloud server stores the data correctly, without any breaks on data.
3) To make sure that TPA will not derive the data of the clients from the information collected in the process.
4) For the secure and efficient transferring of data, and to help with multiple auditing delegations for the huge number of users from different places simultaneously.

III. THE PROPOSED STRATEGY

This section gives the solution for the data and also for the integrity. Firstly, we see about the overview of the process by its definitions. Here we are having 3 algorithms they are for key generation, signature generation and verification. The key generation is run by the client to start the process. Signature generation is made to verify metadata. This privacy preserving is having 2 phases. One is a setup phase where the data from the user is send to the TPA and the TPA checks the data whether from the authorized user or not. The public key is send to the user once he enters the username and password correctly. Then he needs to enter the key in the space provided by the server. Then he can enter into his page where he can send or download the files. Before he downloads the file or uploads, the user sends a request to the server. The second phase is that the TPA will receive the request and sends it to the server. Then server will generate a proof to the TPA who verifies and sends the data to the user. Then the user views the data and downloads the file. In this, the TPA is assumed to be stateless.

3.1 Basic strategy

There are 2 basic schemes, before we write them DSA algorithms. The first one is MAC algorithm and the second is user based on HLA (homomorphic linear authenticators). There are two ways to make use of MAC to authenticate the data. A first way is just uploading the data blocks with their MACs to the server, and sends the corresponding secret key to the TPA. Later, the TPA can randomly retrieve blocks with their MACs and check the correctness. Apart from the communication and complexities, the TPA also requires the knowledge of the data blocks for verification. To get the requirement of the data in TPA verification, one may restrict the verification to just consist of equality checking. However, it suffers from the following severe drawbacks:

1) The more number of times a particular data file can be verified is limited by the number of secret keys that must are numbered already. Once all possible secret keys are used, the user then has to retrieve data in full to again compute and publish new MACs to the TPA.

2) The TPA also has to maintain and updates between each audits that is to check the keys always. Also to consider the large number of audit delegations from different users, maintaining such states for TPA can be difficult and also possibility for the error is high.
3) It can only support static data, and cannot efficiently deal with dynamic data at all [3]. However, supporting dynamic data is most important for cloud storage systems. For the reason of timing and clarity, our main strategy will be based on static data.

The second one we have is HLA based output is, to effectively support without having to gain the data blocks by themselves, the HLA technique is also some enforceable verifying the metadata that authenticate the integrity of a data. The difference is that HLAs can be aggregated. It is possible to execute an aggregated HLA which authenticates a parallel combination of each data blocks. Though it is allowing efficient data auditing, consuming constant bandwidth, the direct adoption of the HLA-based techniques is still not suitable for general purposes. This is because the linear combination of blocks may potentially reveal the user about data information to TPA, and privacy preserving may not be guaranteed. Specifically, if a limited number of the combinations of the same blocks are collected, then the TPA can simply derive the user's data content by solving a system of linear equations.

3.2 proposed scheme
Now, here we present our algorithm of DSA implementation which is helpful for the process of security and integrity. The first one we see is the key generation of the process. The key generation is again having 2 phases. The first phase is that, it generates the parameters.

A. Parameter generation:
Choose an approved cryptographic hash function H. In the actual DSS, H was always SHA-1, at the same time the stronger SHA-2 hash functions are approved for use in the current DSS [4].The hash output may be truncated to the size of a pair of keys, deciding on a key length L and N.
This is the initial measure of the cryptographic strength of the key. The actual DSS derived L to be a multiple of 64 between 512 and 1024 (inclusive). NIST 800-57 recommends lengths of 2048 (or 3072) for keys with security lifetimes extending beyond 2010 (or 2030), using corresponding long N[8].
FIPS 186-3 specifies L and N length pairs of (1024,160), (2048,224), (2048,256), and (3072,256).[5]
Choose an N-bit prime q. N must be less than or equal to the hash output length.
Choose an L-bit prime number with modulus p such that p–1 is a multiple of q.
Choose g, a number whose multiplicative order modulo p is q.
This may be done by setting $g = h(p-1)/q \bmod p$ for some arbitrary h ($1 < h < p-1$), and trying again with a different h if the result comes out as 1. Most choices of h will have to be a usable g; commonly h=2 is used. The algorithm parameters (p, q, g) may be shared between different users of the system.

B. Key generation
Given a set of parameters, the second phase computes private and public keys for a single user:
Choose x by some random method, where $0 < x < q$.
Calculate $y = gx \bmod p$.
Public key is (p, q, g, y). Private Key is x.
There exist an efficient algorithm for computing the modular exponentiations $h(p-1)/q \bmod p$ and
$gx \bmod p$, such as exponentiation by squaring. As now, we generated the keys the process goes with the signature development so that the data can be secured and also that the integration is possible and also that a secret number is also generated if necessary.

C. Secret number generation
A new secret random number k shall be generated prior to the generation of each digital signature for use during the signature generation process.
This secret number shall be protected from unauthorized disclosure and modification. −1 k is the multiplicative inverse of k with respect to multiplication modulo q; i.e., 0 < −1 k < q and 1 = (−1 k k) mod q.
This inverse is required for the signature generation process. A technique is provided in Appendix C.1 for deriving −1 k from k. k and −1 k may be pre-computed, since knowledge of the message to be signed is not required for the computations.
When k and −1 k are pre-computed, their integrity and confidentiality will be protected.

D. Signature generation
Let N be the bit length of q.
Let us assume that, min(N, outlen) denote the minimum of the positive integers N and outlen, where outlen is the bit length of the hash function output block.
The signature of a message M consists of the pair of numbers r and s that is computed according to the following equations:
r = (gk mod p) mod q.

z = the leftmost min(N, outlen) bits of Hash(M).
s = (−1 k (z + xr)) mod q.
While computing s, the string z obtained from Hash (M) shall be converted to an integer. Note that r may be computed whenever k, p, q and g are available, e.g., whenever the domain parameters p, q and g are known, and k has been pre-computed r may also be pre-computed, since knowledge of the message to be signed is not required for the computation of r. Pre-computed k, k-1 and r values shall be protected in the same manner as the private key x until s has been computed (see SP 800-57). The values of r and s shall be checked to determine if r = 0 or s = 0. If either r = 0 or s = 0, a new value of k shall be generated, and the signature shall be recalculated. It is extremely unlikely that r = 0 or s = 0 if signatures are generated properly. The signature (r, s) may be transmitted along with the message to the verifier.
Now the following is the verification and validation of the algorithm.

E. Verification and validation

Signature verification may be performed by any party using the public key. A user may wish to verify that the signature is correct, perhaps before sending the signed message to the intended recipient. The intended recipient (or any other party) verifies the signature to determine its authenticity.
Prior to verifying the signature of a signed message, the claimed signatory's public key, the domain parameters and identity **shall** be made available to the verifier in an authenticated manner. The public key may be obtained in the way of a certificate signed by a trusted entity (e.g., a CA) or in a face-to-face meeting with the public key owner.
Let *M* , *r*, and *s* be the received versions of *M*, *r*, and *s*, respectively; let *y* be the public key of the claimed signatory; and let *N* be the bit length of *q*. Also, let **min**(*N*, *outlen*) denote the minimum of the positive integers *N* and *outlen*, where *outlen* is the bit length of the hash function output block. The signature verification process is as follows:
1. The verifier **shall** check that $0 < r' < q$ and $0 < s' < q$; if either condition is violated, the signature **shall** be rejected as invalid.
2. If the two conditions in step 1 are satisfied, the verifier computes the following:
w = (*s*′)–1 mod *q*.
z = the leftmost **min**(*N*, *outlen*) bits of **Hash**(*M*′).
*u*1 = (*zw*) mod *q*.
*u*2 = ((*r*′)*w*) mod *q*.
v = (((*g*)*u*1 (*y*)*u*2) mod *p*) mod *q*.
The string *z* obtained from **Hash**(*M*') **shall** be converted to an integer. 3.
If *v* = *r*, then the signature is verified. For a proof that *v* = *r* when *M*'= *M*, *r*'= *r*, and *s*= *s*, If *v* does not equal *r*', then the message or the signature may have been modified, there may have been an error in the signatory's generation process, or an imposter (who did not know the private key associated with the public key of the claimed signatory) may have attempted to forge the signature.
The signature **shall** be considered invalid. No inference can be made as to whether the data is valid, only that when using the public key to verify the signature, the signature is incorrect for that data.
As above mentioned, if M′ = M, r′ = r and s′ = s in the signature verification, then v = r′. Let Hash be an approved hash function. The following result is needed.
Lemma: Let p and q be primes such that q divides (p – 1), let h be a positive integer less than p, and let g = (h(p–1)/qmod p). Then (gq mod p) = 1, and if (m mod q) = (n mod q),then (gm mod p) = (gn mod p).
Proof:
gq mod p = (h(p–1) / q mod p)
q mod p = h(p – 1) mod p = 1
by Fermat's Little Theorem[6]. Now let (m mod q) = (n mod q), i.e., m = (n + kq) for some integer k. Then,
gm mod p = gn + kq mod p = (gngkq) mod p = ((gn mod p) (gq mod p)k) mod p = gn mod p, since (gq mod p) = 1.
Proof of the main result:
Theorem: If M′ = M, r′ = r, and s′ = s in the signature verification, then v = r′.
Proof:
w = (s′)–1 mod q = s–1mod q
u1 = ((Hash(M ′))w) mod q = ((Hash(M))w) mod q
u2 = ((r′)w) mod q = (rw) mod q.
Now y = (gxmod p), so that by the lemma,
v = ((gu1yu2) mod p) mod q = ((gHash(M)w yrw) mod p) mod q= ((gHash(M)w gxrw) mod p) mod q = ((g(Hash(M) + xr)w) mod p) mod q.
Also:
s = (k–1 (Hash(M) + xr)) mod q.
Hence:
w = (k (Hash(M) + xr)–1) mod q (Hash(M) + xr)w mod q = k mod q.
Thus, by the lemma:
v = (gk mod p) mod q = r

F. Properties of our proposed scheme

Here, by using this algorithm the process can be easy to the user and also for the organization or the service provider. It is easy to say that this can achieve the privacy of the data and security. Here the user gets the public key and then sends the data or access the data from the server.

IV. CONCLUSION

The final conclusion of the paper is that by using the DSA algorithms the process of the securing the data is been easy and also that it helps the user and also the organization in reducing the time spend o the system or waiting for the response. Here the future work will be developing the system for multiple users and also with greater security. Also, that the TPA present between the processes will not learn anything about the data that was used by the server or the user.

REFERENCES

[1] A. Juels and J. Burton S. Kaliski, "Pors: Proofs of retrievability for large files," in *Proc. of CCS'07*, Alexandria, VA, October 2007, pp. 584–597.

[2] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan," Auditing to keep online storage services honest," in *Proc. Of HotOS'07*. Berkeley, CA, USA: USENIX Association, 2007, pp.1–6.

[3] 104th United States Congress, "Health Insurance Portability and Accountability Act of 1996 (HIPPA)," Online at http://aspe.hhs.gov/admnsimp/pl104191.htm, 1996.

[4] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained access control in cloud computing," in *Proc. of IEEE INFOCOM'10*, San Diego, CA, USA, March 2010.

[5] J. Kincaid, "MediaMax/TheLinkup Closes Its Doors," Online at http://www.techcrunch.com/2008/07/10/mediamaxthelinkup-closes-its-doors/, July 2008.

[6] Cloud Security Alliance, "Security guidance for critical areas of focus in cloud computing," 2009, http://www.cloudsecurityalliance.org.

[7] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. of Asiacrypt 2008*, vol. 5350, Dec 2008, pp. 90–107.

[8] 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, "Above the clouds: A berkeley view of cloud computing," University of California, Berkeley, Tech. Rep.UCB-EECS-2009-28, Feb 2009.