DELEGATED ACCESS CONTROL IN PUBLIC CLOUDS USING TWO LAYER ENCRYPTION

By

MALATHI P

PG Scholar, Muthayammal College of Engineering, Rasipuram, Nammakkal (DT), Tamilnadu, India.

ABSTRACT

A cloud is a collection of terminals and servers that are publicly accessible via the internet. One of the primary uses of cloud computing is data storage. In cloud computing, data is stored in encrypted form to ensure confidentiality. Here the user performs verification during data storage process. So the data owner requires a Third Party Auditor [TPA] for auditing. TPA audits the files stored in the cloud. During the audit process, TPA gets the keys from the data owner and views the data for the auditing. The major drawback here is TPA can modify the owner data. So, the proposed system implements the encrypting data and notification method, so that the TPA views the data in encrypted form. For encrypted data auditing, dynamic Provable Data Possession (PDP) is used. But if a TPA tries to decrypt the owner data, then Provable Of Retrievable (POR) sends the notification to the data owner. Until the owner verifies the notification, the modification will not be committed to the database.

Keywords: TPA Auditing, Provable Data Possession, Provable of Retrievable, Notification.

INTRODUCTION

In cloud storage, user data is stored in the cloud service provider. Security and privacy represent major concerns in the adoption of cloud technologies for data storage. In cloud computing environment, the main advantage is using a Third Party Auditor (TPA)[7]. That third party auditor will audit the cloud environment. Auditing will be performed in two ways. First is the CSP (Cloud Service Provider) side and another is the user side[3]. In CSP side, auditing is done through resource access and maintains scheduling security. In server side, auditing process is space auditing, resource/ Data integrity, Security, storing/ retrieving, Data analysis. In this paper, the user side auditing is carried out, especially the data integrity. One way for increasing data integrity is encrypting the data and providing access control policies[1]. Basically, cloud storage uses single encryption. For more confidentiality, two layer encryption is implemented. A challenging issue in the TLE approach is how to decompose the ACPs[8] so that fine-grained ABAC enforcement can be delegated to the cloud while at the same time the privacy of the identity attributes of the users and confidentiality of the data are assured. In order to delegate as much access control enforcement as possible to the cloud, one needs to decompose the ACPs such that the data owner manages minimum number of attribute conditions in those ACPs that assure the confidentiality of data from the cloud. Each ACP should be decomposed into two sub ACPs such that the conjunction of the two sub ACPs result in the original ACP. The two layer encryption should be performed in such a way that the data owner first encrypts the data based on one set of sub ACPs and the cloud re-encrypts the encrypted data using the other set of ACPs. The two encryptions together enforce the ACP as users should perform two decryptions to access the data. For example, if the ACP is $(C1 \wedge C2) \vee (C1 \wedge C3)$, then the ACP can be decomposed as two sub ACPs C1 and C2 v C3. Notice that the decomposition is consistent; that is, $(C1 \wedge C2) \vee (C1 \wedge C3) = C1 \wedge (C2 \vee C3)$. The data owner enforces the former by encrypting the data for the users satisfying the former and the cloud enforces the latter by re-encrypting the data owner, encrypted data for the users satisfying the latter. Since the cloud does not handle C1, it cannot decrypt the owner encrypted data and thus confidentiality is preserved. Notice that users should satisfy the original ACP to access the data by performing

two decryptions. In this paper, the problem of decomposing is shown by the ACPs such that the data owner manages the minimum number of attribute conditions, while at the same time assuring the confidentiality of the data in the cloud is NP-complete. Two optimization algorithms are proposed to find the near optimal set of attribute conditions and decompose each ACP into two sub ACPs. So, on the auditing process, third party auditor gets only the encrypted data. For the encrypted file auditing, Provable Data Possession (PDP) is implemented[4].

Identity token issuance

IdPs issue identity tokens to Users based on their identity attributes.

Policy decomposition

The owner decomposes each ACP into atmost two sub ACPs such that the owner enforces the minimum number of attributes to assure confidentiality of data from the cloud. It is important to make sure that the decomposed

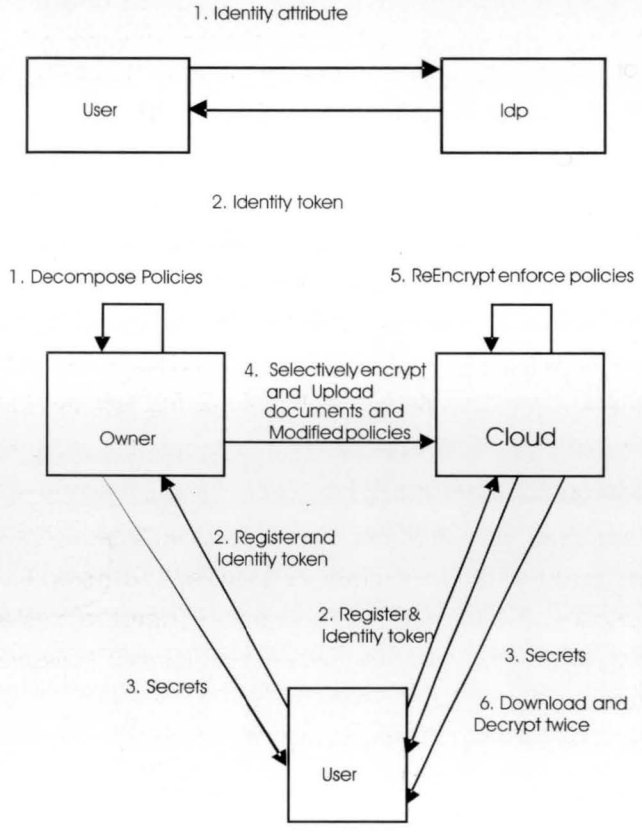

Figure 1. Two Layer Encryption approach

ACPs are consistent so that the sub ACPs together enforce the original ACPs. The owner enforces the confidentiality related sub ACPs and the cloud enforces the remaining sub ACPs.

Identity token registration

Users register their identity tokens in order to obtain secrets to decrypt the data that they are allowed to access. Users register only those identity tokens related to the owner's sub ACPs and register the remaining identity tokens with the cloud in a privacy preserving manner. It should be noted that the cloud does not learn the identity attributes of users during this phase.

Data encryption and uploading

The owner first encrypts the data based on the owner's sub ACPs in order to hide the content from the cloud and then uploads them along with the public information generated by the AB-GKM::KeyGen algorithm and the remaining sub ACPs to the cloud. The cloud in turn encrypts the data based on the keys generated using its own AB-GKM::KeyGen algorithm. Note that the AB-GKM::KeyGen at the cloud takes the secrets issued to users and the sub ACPs given by the owner into consideration to generate keys.

Data downloading and decryption

Users download encrypted data from the cloud and decrypt the data using the derived keys. Users decrypt twice to first remove the encryption layer added by the cloud and then by the owner. As access control is enforced through encryption, users can decrypt only those data for which they have valid secrets.

Encryption evolution management

Over time, either ACPs or user credentials may change. Further, already encrypted data may go through frequent updates. In such situations, data already encrypted must be re-encrypted with a new key. As the cloud performs the access control enforcing encryption, it simply re-encrypts the affected data without the intervention of the owner.

1. Dynamic Provable Data Possession (DPDP)

In this paper, the two layer encryption (Figure 1) is implemented, which means that the TPA gets encrypted

form of data for auditing. For encrypting data auditing, TPA uses the Provable Data Possession (PDP) [2]. In that, PDP has two phases: setup phase and verification phase

1.1 Notations

- D – Out sourced data. We assume that D can be represented as a single contiguous file of d equal sized blocks: D [1];:::; D[d]. The actual bit-length of a block is not germane to the scheme.
- OWN – the owner of the data.
- SRV – the server, i.e., the entity that stores outsourced data on owner's behalf.
- H(_) – cryptographic hash function. In practice, we use standard hash functions, such as SHA-1, SHA-2, etc.
- AE key(_) – an authenticated encryption scheme, that provides both privacy and authenticity. In practice, privacy and authenticity is achieved by encrypting the message first and then computing a Message Authentication Code (MAC) on the result. However, a less expensive alternative is to use a mode of operation for the cipher that provides authenticity in addition to privacy in a single pass, such as OCB, XCBC, IAPM.
- Fkey (_) – Pseudo-Random Function (PRF) indexed on some (usually secret) key. In practice, a "good" block cipher acts as a PRF, in particular. We could use AES in which keys, input blocks, and outputs are all of 128 bits (AES is a good pseudo-random permutation). Other even more practical constructions of PRFs deployed in standards use "good" MAC functions, such as HMAC [5].
- gkey(_)-Pseudo-Random Permutation (PRP) indexed under key. AES is assumed to be a good PRP.

1.2 Setup Phase

Starting with a database D and dividing into d blocks, the storage server is challenged t times. A pseudo random function, f is used and a pseudorandom permutation, g is defined as:

f : {0;1}c *{0;1}k {0;1}L

and

g : {0;1}l _{0;1}L {0;1}l :

In this case, l = logd since g is used to permute indices. The output of f is used to generate the key for g and c = log t. It is noted that both f and g can be built out of standard block ciphers, such as AES (Advanced Encryption Standard). In this case L =128. We use the PRF f with two master secret keys W and Z, both of k bits[6]. The key W is used to generate session permutation keys, while Z is used to generate challenge nonces. During the setup phase, the owner OWN generates in advance t possible random challenges and the corresponding answers. These answers are called tokens. To produce the i^{th} token, the owner generates a set of r indices as follows:

- First, generate a permutation key $k_i = f_w(i)$ and a challenge nonce $c_i = f_z(i)$.
- Then, compute the set of indices: $fl_j 2 [1; : : : ;d] j 1 \_ j \_ rg$, where $l_j = gk_i(j)$.
- Finally, compute the token:

  $v_i = H(c_i; D[l1]; : : : ; D[l_r])$.

Basically, each token v_i is the answer we expect to receive from the storage server whenever we challenge it on the randomly selected data blocks $D[l1]; : : : ; D[l_r]$.

The challenge nonce c_i is needed to prevent potential pre-computations performed by the storage server. Notice that each token is the output of a cryptographic hash function, so its size is small. Once all tokens are computed, the owner outsources the entire set to the server, along with the file D, by encrypting each token with an authenticated encryption function2. The setup phase is shown in detail in Algorithm 1.

1.3 Verification Phase

To perform the i-th proof of possession verification, OWN begins by re-generating the i-th token key k_i as in step 1 of Algorithm 1. Note that OWN only needs to store the master keys W;Z; and K, plus the current token index i. It also re-computes c_i as above. Then

Algorithm 1: Setup phase begin

Choose parameters c; l; k; L and functions f ; g;

Choose the number t of tokens;

Choose the number r of indices per verification;

Generate randomly master keys

W; Z; K 2 {0; 1}^{k}. for (i 1 to t) do begin Round i

Generate $k_i = f_W(i)$ and $c_i = f_Z(i)$

Compute

$v_i = H(c_i; D[g_{ki}(1)]; :::; D[g_{ki}®)])$

Compute $v^0_i = AE_K(i; v)$ end

Send to SRV : {D; f[i; v^0] for 1 i tg) end

OWN sends SRV both k_i and c_i (step 2 in Algorithm 2). Having received the message from OWN , SRV computes:

$z = H(c_i; D[g_{ki}(1)]; :::; D[g_{ki}®)])$

SRV then retrieves v^0_i and returns [z; v^0_i] to OW N who, in turn, computes $v = AE_K^{-1}(v^0_i)$ and checks whether v=(i, z). If the check succeeds, OWN assumes that SRV is storing all of D with a certain probability.

Algorithm 2: Verification phase

begin Challenge i

OW N computes $k_i = f_W(i)$ and $c_i = f_Z$

OWN sends fk; c$_i$g to SRV

SRV computes

$z = H(c_i; D[g_{ki}(1)]; :::; D[g_{ki}®)])$

SRV sends fz; v^0_ig to OWN

OW N extracts v from v^0_i. If decryption fails or v 6= (i ; z) then REJECT.

End

It is pointed out that there is almost no cost for OWN to perform a verification. It only needs to re-generate the appropriate [k_i; c_i] pair (two PRF-s invocations) and perform one decryption in order to check the reply from SRV . Furthermore, the bandwidth consumed by the verification phase is constant (in both step 2 and 4 of Algorithm 2). This represents truly minimal overhead. The computation cost for SRV , though slightly higher (r PRP-s on short inputs, and one hash), is still very reasonable.

1.4 Notification

Considering the probability Pesc that OWN manages one run is successfully completed in the verification protocol, while SRV has either been deleted, or tampered with m data blocks. Note that SRV avoids detection only if none among all the r data blocks involved in the i-th token verification process, are deleted or modified.

Pesc = (1-(m/d)) r

For example, if the fraction of corrupted blocks (m=d) is 1% and r = 512, then the probability of avoiding detection is below 0:6%.

Considering economically-motivated adversaries that are rewilling to modify or delete, a percentage of the file is stressed. In this context, deleting or modifying few bits won't provide any financial benefit. If detection of any modification/deletion of small parts of the file is important, then erasure codes could be used. The file is first encoded and then the PDP protocol is run on the encoded file. Erasure codes are used in the context for checking storage integrity. Erasure codes can help in detecting small changes to the file and possibly recover the entire file from a subset of its blocks. However, erasure codes (even the near-optimal ones) are very costly and impractical for files with a very large number of blocks (the type of files that are considered here). In addition, encoding will significantly expand the original file and render impractical operations on dynamic files, such as updating, deleting, and appending file blocks. Notice that the discussion above concerns erasure codes applied to the file by the client and not by the server. Obviously, the server may (and should) use erasure codes independently to prevent data corruptions resulting from unreliable storage.

1.5 Security Analysis

Security definitions are followed in particular, the one which is presented in the form of a security game. In practice, it is required to prove that the protocol is a type of proof of knowledge of the queried blocks, i.e., if the adversary passes the verification phase, then one can extract the queried blocks. Another formal definition for the related concept of sub-linear authentication is provided.

There is a challenger that plays a security game (experiment) with the adversary A. There is a setup phase where A is allowed to select data blocks D[i] for 1 _ i _ d. In the verification phase, the challenger selects a nonce and r random indices and sends them to A. Then the

adversary generates a proof of possession P for the blocks queried by the challenger. If P passes the verification, then the adversary wins the game. This Provable Data Possession scheme states that, if for any probabilistic polynomial-time adversary A, the probability that A wins the PDP security game on a set of blocks is negligibly close to the probability that the challenger can extract those blocks. In this type of proof of knowledge, a "knowledge extractor" can extract the blocks from A via several queries, even by rewinding the adversary which does not keep state. At first, it may seem that it is not needed to model the hash function as a random oracle and just a collision-resistance property (assuming we are using a "good" standard hash unction and not some contrived hash cheme) is required. However, the security definition for PDP is an extraction-type definition since it is required to extract the blocks that were challenged. So the random oracle model should be used but not really in a fundamental way. The ability to see the inputs to the random oracle is not using its programmability feature (i.e., we are not "cookingup" values as outputs of the random oracle). Based on the inputs, the queried data blocks are extracted. Since only the random oracle is used, the proof does not rely on any cryptographic assumption, but it is information-theoretic.

Theorem 1

This scheme is a secure Provable Data [9] Possession scheme in the random oracle model.

Proof sketch

Assume that AEK(_) is an ideal authenticated encryption. This implies that, given AEK(X), the adversary cannot see or alter X, thus assume that X is stored directly by the challenger (i.e., there is no need for the adversary to send X to the challenger and we can remove AEK(X) from the picture). More formally, the game is equivalent to the following:

- A simulator S sets up a PDP system and chooses its security parameters.
- The adversary A selects values x1, : : :, xn and sends them to the simulator S
- The adversary can query the random oracle at any point of time. For each input to the random oracle, the simulator replies with a random value it stores the input and the corresponding output in a table (to be consistent with the replies).
- At the challenge phase, the simulator challenges A on the i th value, x_i and sends a random value c_i to A.
- A replies with a string P.

Note that, in the original game, the value x_i corresponds to the ordered sequence of r concatenated blocks which are queried during the i^{th} challenge. In addition,the simulator can query any x_i only once.

Clearly, if the adversary wins the game ($P=H(c_i; x_i)$) with non-negligible probability, then x_i is extracted with non-negligible probability. This is because the best the adversary can do to impede the extraction of xi is to guess the correct output of the random oracle or to find a collision (a random oracle is trivially collision-resistant). Both these events happen with negligible probability.

2. Multi Cloud

The user data will separate as multi parts and stored in multi-clouds for being cost-effective. In cloud, storage cost is based on the size of the data. If big data are stored in one cloud, then the cost will be high. So to avoid this, data is separated as small parts and stored in multi-clouds. In this way, one cloud can store a smalll size of the data so that the cost is effective.

Conclusion

Current approach has to enforce ACPs on outsourced data using selective encryption that require organizations to manage all keys and encryptions and upload the encrypted data to the remote storage. Such approaches incur high communication and computation cost to manage keys and encryptions whenever user credentials or organizational authorization policies/data change. Then if attacker attacks the cloud service provider, then the encryption file will be modified or attack the original information. So the security policies will be affected.

In this paper, a two layer encryption based approach is proposed to solve this problem by delegating as much of the access control enforcement responsibilities as possible to the cloud, while minimizing the information

exposure risks due to colliding users and cloud. A key problem in this regard is how to decompose ACPs so that the owner has to handle a minimum number of attribute conditions while hiding the content from the cloud. The policy decomposition problem is NP-Complete and provides approximation algorithms. Based on the decomposed ACPs, a novel approach is proposed for privacy preserving fine grained delegated access control to data in public clouds. This approach is based on a privacy preserving attribute based key management scheme that protects the privacy of users while enforcing attribute based ACPs. As the experimental results show, decomposing the ACPs and utilizing the two layer of encryption reduces the over head at the owner. Then The security of cloud service provider will be increased. So, the confidentiality of the data and encrypted file will be high. In that, if a third party auditor attempts to modify the content, the client receives the notification. Until the user verification is completed, the modification will not be committed to the cloud storage. In this way, the attacker modifying the content can be reduced and then confidentiality can be increased.

References

[1]. **Alina Oprea, Michael K. Reiter,Ke Yang (2005).** "Space-efficient block storage integrity", *In Proc of NDSS on 2005.*

[2]. **C. C. Erway, A. Kupc¸ U, C. Papamanthou, and R. Tamassia (2011).** "An improved dynamic provable data possession model" *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on,* Sept. 2011.

[3]. **Cong Wang, Qian Wang, and Kui Ren (2009).** "Ensuring data storage security in Cloud Computing" *Quality of Service IWQoS 17th International Workshop* on, July 2009.

[4]. **G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik (2013).** "Scalable and efficient provable data possession" *Secure Comm Security and Privacy in Communication Networks*, June 2013, Volume 18, Issue 3, pp. 265-271.

[5]. **Hovav Shacham, Brent Waters (2013).** "Compact proofs of retrievability", *Journal of Cryptology,* July 2013, Volume 26, Issue 3, pp 442-483.

[6]. **Junkil Park,Jin-Young choi (2007).** "Formal Security Policy Model For Commaon Criteria Evaluation" Advanced Communication Technology. *The 9th International Conference* (Volume,1), Feb 2007.

[7]. **Mehul A. Shah Ram Swaminathan Mary Baker (2008).** *'Privacy-preserving audit and extraction of digital contents', Hewlett-Packard,* on Apr 2008.

[8]. **Mohamed Nabeel, Ellisa Bertino (2013).** "Privacy preserving delegated access control in public clouds" *IEEE International Conference,* 2013.

[9]. **Qian Wang, Kui Ren ,Wenjing Lou ,Yanchao Zhang (2009).** "Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance" *INFOCOM 2009, IEEE,* April 2009.

ABOUT THE AUTHOR

PG Scholar, Muthayammal College of Engineering, Rasipuram, Nammakkal (DT), Tamilnadu, India.