A Secure Access Code Technique for Remote Data Integrity on Public Cloud

Karthik. P
Department of CSE,
WPTC, PIPMATE
Pondicherry, India

Krishna Kumar. V
Department of CSE,
KPTC, PIPMATE
Karaikal, India

ABSTRACT
The present business trend highly demands more sophisticated techniques for remote data storage to stay alive in the competition among the business holders. The situation is more critical when the data to be stored is confidential information such as business policies, business rules and statistical data about the organization that is often needed to be shared with in the trusted enclave. Inevitably this scenario forces the need for the remote data integrity on mistrusted public cloud where the integrity of the data is not only being checked by the owner of the file but also by the people in the trusted enclave. This paper addresses new solution for remote data integrity using access code by implementing data level dynamics without the need for third party auditors.

Keywords
Remote Data Integrity, Access Code, Cloud Computing, Public Cloud.

1. INTRODUCTION
In the world of today's computing, computational intensive applications occupies more space. This makes everyone in the universe to move towards cloud computing. The major reasons include; less infrastructure cost (hardware, software, application and operating system), better performance, less maintenance, unlimited storage capacity, increased computing power, increased data safety and any time any where accessibility.

Figure 1: Typical remote data Storage on cloud

The primary motivation behind the remote data storage in the public cloud is minimizing the investment cost and maximizing the benefit without compromise in security. Cloud storage is not without challenges. The biggest problem in dealing with the remote data storage is the data integrity. This may be caused due to illegal access/modification of remote data by any authorized persons within the trusted enclave, third party verifiers, anonymous persons or any malicious servers. The integrity of the remote data if unnoticed for a long time or not dealt with care may cause serious problems such as; false data access by the clients, complete data loss and as the worst case the entire business loss.

C. Wang 'etal, [9],[10] proposed solution for user data privacy with third party auditor. The integrity issues of remote data with block level data dynamics are discussed in [1],[2],[4],[5],[6]. The problems associated with public verifiability are dealt in [7],[8],[9],[10],[14]. Privacy against third party verifiers is achieved through cryptographic algorithm [12] as suggested in [11],[13].

The protocols [1],[2],[4],[5] have suggested the following guidelines for the remote data integrity solutions: elimination of redundant copies of the files, security against the malicious server, less communication and computational overhead without compromising the security, exclusion of third party auditors and public verifiability.

2. CONTRIBUTION OF THE PAPER
This paper proposes a novel access-code technique for remote data integrity on public cloud with data level dynamics. The proposed protocol enables the client or the trusted enclave to perform data dynamics at byte level. The modification of remote data due to data dynamics results in change of content. This creates a problem of regeneration of pre computed tags for the entire file which results computation overhead for the server. This protocol resolves this problem by allowing the client/trusted enclave to perform data dynamics at byte level and calculate pre computed tags only for the modified content. The size of tag is very small and their transmission will not create problems associated with network bandwidth and congestion.

The merits of the proposed protocol are: Content security against third party auditors, Public verifiability, Data level dynamics with less computational overhead, Complete access control over remote data, Prevention of unauthorized access with access-code.

The paper is organized as follows. Part-3 discusses about remote data integrity, Part-4 deals with Data Level dynamics

using access code, Part-5 discuss about efficiency and comparison of the proposed protocol with existing protocols mentions, Part-6 concludes with the future work of the proposed protocol. The proposed protocol suggests efficient techniques in support of data dynamics at data level. The paper combines the merits of block level dynamics with data level dynamics in order to ensure better turnaround time for the client while conducting verification test. Whenever a piece of data is modified, the corresponding block will be automatically identified and the verification tags will be automatically updated only for the modified blocks.

The two major components of the proposed protocol are: Remote Data Integrity and Data Level Dynamics.

3. REMOTE DATA INTEGRITY

Remote data integrity enables the client to perform public verifiability on public cloud data any number of times without communication and network bandwidth issues. This involves the following major modules.

3.1 Key Generation:

In this section, the algorithm listed in 3.1.1 generates two keys which can be used for user authentication and public verifiability.

3.1.1 Algorithm KeyGen

It accepts ***K*** as random seed and it returns two keys namely ***pk*** and ***sk***. Here ***f1(x)*** and ***f2(x)*** are non-linear floating point functions.

[pk, sk] = KeyGen(K)

Begin

Step 1: Split the Seed K into two parts K1 and K2

Step 2: pk = f1(K1)

sk = f2(K2)

End

3.2 Tag Generation:

This module involves creation of tags using the keys obtained from the previous section 3.1. It generates two types of tags; client tag and server tag using the algorithm listed in 3.2.1. The client tag is a unique code given to the trusted enclave that enables them to perform public verifiability. The server tag is compared with the client tag to perform authentication.

3.2.1 Algorithm GetTags

It accepts three parameters: the keys ***pk*** and ***sk*** and the file ***m***. It returns two tags; client tag $\boldsymbol{D_c}$ and server tag $\boldsymbol{D_m}$.

$[D_c, D_m]$ = GetTags(pk, sk, m)

Begin

Step 1: Split the file ***m*** into ***n*** equal blocks such that; $\boldsymbol{m = m_1, m_2, m_3 \ldots m_n}$; where, the length of the each file block is denoted by l

Step 2: Generate a unique numeric tag ***Di*** for every file block $\boldsymbol{m_i}$; denoted as $<m_1, m_2, \ldots, m_n> = <D_1, D_2, D_3 \ldots D_n>$.

Step 3: For every $\boldsymbol{D_i}$ two arbitrary constants $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ are chosen such that $\boldsymbol{C_1 + C_2 = D_i}$.

Step 4: Calculate $\boldsymbol{U_i = \{C_{i1}, C_{i2}\}}$ where $\boldsymbol{C_{i1}}$ and $\boldsymbol{C_{i2}}$ are obtained by the encryption of $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ using encryption algorithm [12].

Step 5: Calculates the server tag $\boldsymbol{D_m}$ as; $\boldsymbol{D_m = \{U_1, U_2, U_3, \ldots, U_n\}}$

Step 6: Calculates the client tag $\boldsymbol{D_c}$ as $\boldsymbol{D_c = \{ T_1, T_2, T_3, \ldots, T_n\}}$; where each $\boldsymbol{T_i}$ is calculated from $\boldsymbol{U_i \in D_m}$ such that $\boldsymbol{T_i = C_{i1} + C_{i2}}$, i ∈{ 1..n }

End

The value of the $\mathbf{D_m}$ will be permanently stored in the cloud along with the file and $\mathbf{D_c}$ will be made available to the trusted enclave for enabling them public verifiability.

3.3 Public Verifiability:

This module enables the trusted enclave to check the integrity of a file stored in the public cloud. The algorithm listed in 3.3.1 performs integrity checking using HVT property.

$$\boldsymbol{Tag}(A) + \boldsymbol{Tag}(B) = \boldsymbol{Tag}(A+B) \qquad \text{----- } \boldsymbol{Eqn.(1)}$$

Where $\boldsymbol{Tag}(p) = \boldsymbol{g^p} \bmod \boldsymbol{N}$ and N is the product of two prime numbers and p is the co prime of N.

3.3.1 Algorithm CheckIntegrity

It accepts the client tag Dc and returns ‘pass’ or ‘fail’ as result. $\boldsymbol{D_m}$ is the corresponding server tag for $\boldsymbol{D_c}$.

[‘pass’ , ‘fail’] = CheckIntegrity($\boldsymbol{D_c}$)

Begin

Step 1: Apply HVT property for the tags Dm and Dc. Where,

$\boldsymbol{D_m} = \{ \boldsymbol{U_1, U_2, U_3, \ldots U_n}\}$ and $\boldsymbol{D_c} = \{ \boldsymbol{T_1, T_2, T_3, \ldots, T_n}\}$

$\boldsymbol{Tag}(U_i) = \boldsymbol{Tag}(T_i)$, Therefore by Eqn.(1) ,

$\boldsymbol{g^{Ci1}} \bmod \boldsymbol{N} + \boldsymbol{g^{Ci2}} \bmod \boldsymbol{N} = \boldsymbol{g^{Ti}} \bmod \boldsymbol{N}$

Where $\boldsymbol{U_i \in D_m}$ and $\boldsymbol{U_i = \{C_{i1}, C_{i2}\}}$

Step 2: Perform step1 $\forall \boldsymbol{U_i \in D_m}$

Step 3: If step2 is true $\forall \boldsymbol{U_i \in D_m}$ then return ‘pass’,

Otherwise return ‘fail’.

End

The algorithm returns ***‘pass’*** denotes that the verifier has succeeded in the verification test. Otherwise the verifier has failed the verification test.

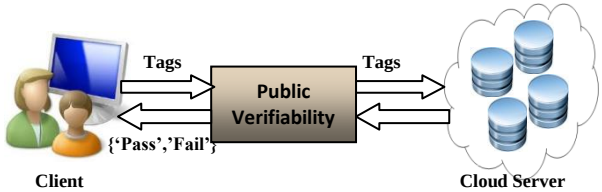

Figure 2: Public Verifiability by trusted enclave

The biggest advantage of this approach is the client or the public in the trusted enclave can call this module any number of times. This will not consume much time. Because it performs integrity checking using the pre computed tags. Therefore the performance of the protocol will not decrease even if the integrity test conducted many times.

4. DATA LEVEL DYNAMICS

The proposed protocol support data dynamics operations on smallest unit of storage called bytes. Therefore the user can perform insert, update, delete and view operations on remote data at byte level.

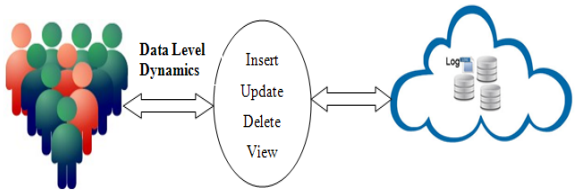

Figure 3: Data level Operations

Here the real challenges to overcome are; Ensuring total control of the file with the owner, Reduce the frequency of tag generation besides allowing the trusted enclave to perform the data dynamics, Minimizing the computation overhead of the cloud server without compromising the security, and enabling the client to restore the data back into the public cloud in the event of illegal data modification by the trusted enclave.

Whenever a piece of data is modified then the corresponding changes are stored permanently in the cloud as log files. These files can only be accessed by the owner of the file. This facility brings power to the owner to have complete control over the remote data even if it is being shared with trusted enclave. Illegal modification or access can be identified and original content can be easily restored back through log files. Further, their access can be blocked by denying permission to them by removing their entry from the trusted enclave.

This facilitates the client/verifiers to modify data into the remote database. Modification can be done only after getting access code from the owner/client of the file. The process requires the following parameters. File ID (***m***), public key (***pk***), position at which the modification is to be done(***pos***), The data String to be changed (***DS***) and Access code(***AC***).

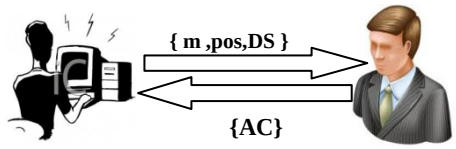

Figure 4: Two way Handshaking to get access code

This access code should be obtained from the client, which will perform authentication sequence for the genuineness of the request. The sequence of operations is depicted as above. The person who wishes to perform modification should inform about the content to be modified to the owner. This is necessarily to be done to hide the structure of the remote file from the requester and also to prevent the remote data is being illegally manipulated by the trusted enclave without the knowledge of the owner. In response to the request the owner will supply two parameters namely ***pos*** and ***AC***. (The mode through which the access code obtained is as similar as that of obtaining ***pk***).

4.1.1 Access Code Generation:

The access code is generated at the client side which is of the form;

$$AC = \{ EX , EY , T \} \quad \text{----- Eqn.(2)}$$

Where,

EX is Encrypted ***X*** and ***EY*** is Encrypted ***Y***

Then the values of ***X*** and ***Y*** are chosen such that

$X, Y \in Z_N$; and $X + Y = T$

And the value of ***T*** is calculated as

$$T = Epos + ED + \zeta \quad \text{----- Eqn.(3)}$$

Where, ***Epos*** is encrypted position value, ***ED*** is the encrypted secret key value and ζ is the encoded value of the data string ***DS***.

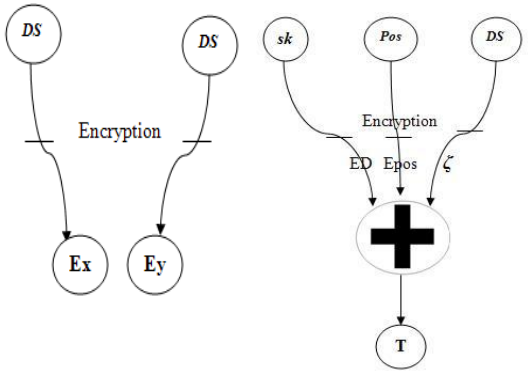

Figure 5: Access Code Generation (AC={E_X, E_Y, T})

4.1.2 Authentication on Cloud Server:

On receiving parameters from the client, the cloud server now computes ***SEpos***(Encrypted value of the received Position parameter) , S_ζ- (Encoded value of the received string) and ***SED*** (Encrypted value of secret key from the server side). Then the server computes ***SX*** and ***SY*** by decrypting EX and EY using the secret key. Finally the this module checks for the equation;

$$SEpos + SED + S_\zeta = T \quad \text{----- Eqn.(4)}$$

$$SX + SY = T \quad \text{----- Eqn.(5)}$$

The server allows the request to modify the remote data only if it satisfies the above equations. Otherwise the request will be simply considered as illegal and denied.

4.1.3 Tag Generation for the Modified File Content:

After successful verification, the cloud server accesses the encrypted content of the remote data using the parameter value ***m***. It calculates the new tag after the data is modified (insert, update, delete) using the algorithm listed below.

4.1.3.1 Algorithm GetModifiedTags

This algorithm generates tags for the modified data after performing data level dynamics at the given position. It accepts the parameters File-id(***m***), keys(***pk*** and ***sk***), position value (***pos***) and the data string ***DS***.

GetModifiedTags (m, pk, sk, pos, DS)

Begin

Step 1: Calculate block number as ***Blkno***=$\lfloor pos/\ l \rfloor$; where ***l*** is the block size (Floor function is used to calculate the block number after where the modified tag should be generated).

Step 2: Calculate ***Offset***= ***pos*** mod ***l***

Step 3: Modify the data (insert, update, delete using the Offset value)

Step 4: Copy the encrypted content of file ***m*** in to a temporary buffer ***Temp1*** as

Temp1= Encrypted content of ***m*** from 0 to ***Blkno**** ***l***

Step 5: Decrypt the content of file ***m*** from ***pos*** to file size ***FS*** and copy it into a temporary buffer ***temp2*** as

Temp2= Decrypted content of ***m*** from ***pos*** to file size ***FS***.

Step 6: Update the file data accordingly using ***Temp1*** and ***Temp2***;

Step 7: Call the algorithm 3.2.1 GetTags with the content of ***Temp2*** to generate tags only for the modified file blocks.

Step 8: Encrypt the contents of ***Temp2*** using ***pk*** and the results are appended with ***Temp1***

End

In the example for insertion given in Figure.6., the protocol does preprocessing by appending null characters to the end of the file in order to make the file size exactly equals to the multiples of block size. Next it will perform encryption[12] on preprocessed data after generating tags for each file block. On insertion the protocol calculates pre-computed tags only for the modified blocks and reduces the computation overhead of the remote server. To perform this operation one has to get access code from the owner of the file. The access code is generated such a way that it will only allow the client / trusted enclave to perform data dynamics as negotiated with the owner of the file.

Any attempt to perform illegal operation will be identified through access code and those requests are simply denied.

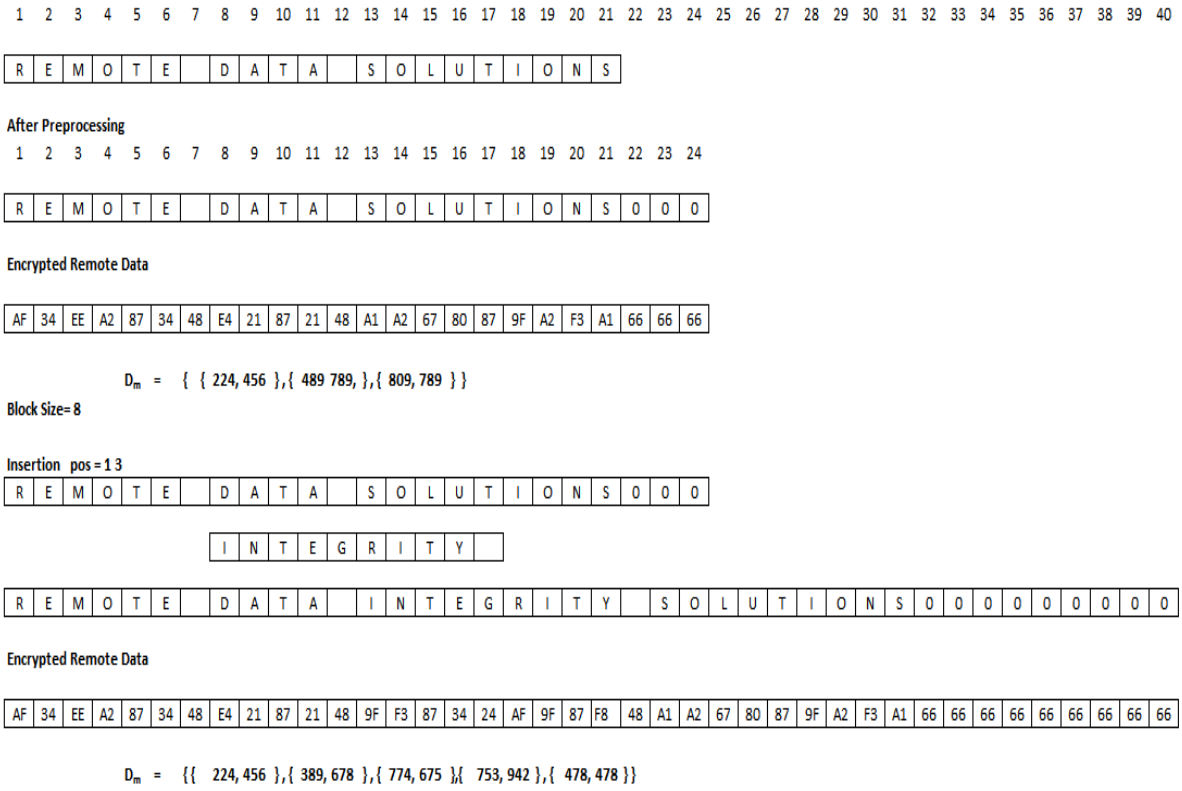

Figure 6: An Illustrative Diagram for data level insertion

The other modules update, delete and view work in similar way and each module is to be executed with access code. But view module slightly differs from others by the way of getting access code from the client. It exchanges two parameters with the client namely ***pos*** and the ***length of the data portion to be seen***.

5. EFFICIENCY AND COMPARISON OF THE PROPOSED PROTOCOL WITH EXISTING PROTOCOLS

The approaches mentioned by ZhuoHao' etal [1], F.Sabe'etal [2] , C.Wang'etal[7] and Z. Hao and N. Yu [14] , require sharing of private keys with the trusted enclave in order to enable them to modify the remote content without affecting the semantics of the remote content. This will pave way for backdoor attack against the security of the remote content.

The Proposed protocol reduces the computation overhead by calculating the tags only for the modified blocks. Hence for the same reason the proposed protocol provides reasonably good performance even for larger files. In addition to the above the protocol forces the trusted enclave to go for negotiation with the client to get access code for performing any kind of data dynamics.

Thus the proposed protocol ensures the integrity of the remote data against third party verifiers, provides complete control to the owner on remote file, supports public

verifiability without computation overhead and finally prevents any backdoor attack by the trusted enclave itself.

The careful analysis of the proposed protocol reveals the fact that it produces better turnaround time for public verifiability without compromising the security. It eliminates the problems of network congestion and bandwidth using pre-computed tags.

S. No	Features	S-PDP	F.Sabe'etal	C.Wang'etal	DPDP	CPDPC	PRDICP	Proposed Protocol
1	Type of Guarantee	Probabilistic / Deterministic	Deterministic	Probabilistic			Deterministic	Deterministic
2	Public Verifiability	Yes	No	Yes	No	Yes	Yes	Yes
3	With help of TPA3	No	No	Yes	No	Yes	No	No
4	Data dynamics	Append Only	Yes	Yes	Yes	Yes	Yes	Yes
5	Privacy Preserving	No	-	No	-	Yes	Yes	Yes
6	Sampling Support	Yes	No	Yes	No	Yes	No	Yes
7	Size of Verification Tag	O(n) *						
8	Communication	O(1)		Oclog(n)		O(s)	O(1)	O(1)
9	Server Block Access	O(c)	O(n)	O(c)			O(n)	O(n)
10	Server Computation	O(c)	O(n)	Oclog(n)		O(c+s)	O(n)	O(n)
11	Verifier Computation	O(c)	O(n)	Oclog(n)		O(c+s)	O(n)	O(n)
12	Client Storage	O(1)	O(n)	O(1)			O(n)	O(n)
13	Data Level Dynamics	No						Yes

Table 1: Comparison of the proposed protocol with existing protocols

6. CONCLUSION AND FUTURE WORK

The proposed protocol provides solution for remote data integrity with data level dynamics and it combines the logic of block level data dynamics for efficiency concerns. It satisfies all the guidelines necessity for efficient remote data integrity. Using the protocol the client can efficiently manage the integrity of remote data while simultaneously allowing the users to perform data level dynamics. The proposed protocol reduces the computation overhead by calculating the tags only for the modified content. In addition to the above the protocol forces the trusted enclave to go for negotiation with the client to get access code for performing any kind of data dynamics. The proposed protocol provides best possible solution for the integrity of remote data files and the research work is going on for providing Integrity solution for all kinds of data including binary files.

7. REFERENCES

[1] Sheng Zhong and Nenghjai Yu, A Privacy- Preserving Remote Data Integrity Checking Protocol with data dynamics and public verifiability, *IEEE transactions on knowledge on Knowledge Engineering,1041-4347/11,2011.*

[2] F.Sebe, J. Domingo-Ferrer, a.Martinez-Balleste, Y.Deswarte, and J.-J.Quisquarter, " Efficient Data Possession Checking in Critical information infrastructures", *IEEE transactions on knowledge on Knowledge Engineering*, Vol 20, pp. 1034-1038, Aug-2008.

[3] http://ebookbrowse.com/advantages-anddisadvantages-of-cloud-computing-pdf-d185401413.

[4] Y.Deswarte,J.-J.Quisquarter, and A.Saidane, " Remote Integrity checking, " *Integrity and internal control in*

information systems VI, pp.1-11. Kluwer academic publishers, Nov.2003.

[5] D.L.Gazzoni-Filho and P.S.Licciardi-Messeder –Barreto, "Demonstrating Data Possession and Un cheatable data Transfer," *Cryptography e-Print Archive*, Report 2007/243,http://eprint.iacr.org/,2006.

[6] C. Erway, A. K¨upc¸ ¨u, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *CCS'09*, pp. 213–222, ACM, 2009.

[7] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in *IWQoS'09*, pp. 1 –9, july 2009.

[8] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *14th ESORICS*, Springer, September 2009.

[9] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy preserving public auditing for data storage security in cloud computing," in *InfoCom2010, IEEE*, March 2010.

[10] C. Wang, S. S.-M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy preserving public auditing for secure cloud storage", Cryptology e-Print Archive, Report 2009/579,2009. http://eprint.iacr.org/.

[11] Y. Zhu, H. Wang, Z. Hu, G.-J.Ahn, H. Hu, and S. S. Yau, "Cooperative provable data possession. Cryptology e-Print Archive, Report 2010/234, 2010. http://eprint.iacr.org/.

[12] Majid Babaei, A novel text and image encryption method based on chaos theory and DNA computing, Natural Computing, March 2013, Volume 12, Issue 1, pp 101-107.

[13] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L Kissner, Z. Peterson, and D. Song, "Provable data possession at un trusted stores," in *CCS'07*, (New York, NY,USA), pp. 598–609, ACM, 2007

[14] Z. Hao and N. Yu, "A multiple-replica remote data possession checking protocol with public verifiability," in *ISDPE2010, IEEE*.