AN EFFICIENT DISTRIBUTION VERIFICATION PROTOCOL (EDVP) FOR DATA STORAGE SECURITY IN CLOUD COMPUTING

[1]N.Sudhir Reddy, [2]J.Sushma, [3]I.Tabitha, [4]Brahmam
[1,2,3,4] Assistant Professor
Department of Computer Science and Engineering
Malla Reddy College of Engineering, Hyderabad.

Abstract -- Cloud Computing (CC) is an emerging computing paradigm that provides large amount of computing and storage to the Clients provisioned as a service over the internet in a pay-as you-go pricing model, where the Clients pay only according to the usage of their services. In this thesis, we investigate this kind of security issues of cloud storage and propose New Probabilistic Efficient and Secure Protocols for data storage security. To avoid integrity availability & confidentiality for cloud storage. To provide better security to the consumers an efficient protocols and methodologies are to be used for cloud in order to store the data with third party members the main problem is security so in my thesis by using EDVP we can provide better security to the customers in cloud.

Keywords – Cloud Computing, Storage, Security, Clients, Service, Protocols, Data.

I. INTRODUCTION

This protocol implements the RSA-DPAP, ECC-DPAP and PVDSSP in a distributed manner which was discussed in chapters 5 and 6 respectively. Here, the n verifiers challenge the n servers uniformly and if m server‘s response is correct then, we can say that Integrity of data is ensured as to verify the Integrity of the data, the verification process uses multiple TPAs. Among the multiple TPAs, one TPA will act as main TPA and remaining are SUBTPAs. The main TPA uses all SUBTPAs to detect data corruptions efficiently, if main TPA fails, the one of the SUBTPA will act as main TPA. The SUBTPAs do not communicate with each other and they would like to verify the Integrity of the stored data in cloud, and the

163 consistency of the provider‘s responses. The propose system guarantees atomic operations to all TPAS; this means that TPA which observe each SUBTPA operations are consistent, in the sense that their own operations, plus those operations whose effects they see, have occurred atomically in same sequence. The Centrally Controlled and Distributed Data paradigm, where all SUBTPAs are The Centrally Controlled and Distributed Data paradigm, where all SUBTPAs are controlled by the TPA and SUBTPA‘s communicate to any Cloud Data Storage Server for verification. We consider a synchronous distributed system with multiple TPAs and Servers. Every SUBTPA is connected to Server through a synchronous reliable channel that delivers challenge to the server. The SUBTPA and the server together are called parties

P. A protocol specifies the behaviors of all parties. An execution of P is a sequence of alternating states and state transitions, called events, which occur according to the specification of the system components. All SUBTPAs follow the protocol; in particular, they do not crash. Every

SUBTPA has some small local trusted memory, which serves to store distribution keys and authentication values. The server might be faulty or malicious and deviate arbitrarily from the protocol such behavior is also called

Byzantine failure. A party P that does not fail in an execution is correct.

II. APPROACH

Here, the Coordinator will randomly generates a bit string for each SUBTPA termed as Task Distribution Key (TDK). Each SUBTPA will successively apply their TDK on the generated Sobol sequence as a mask up to the sequence will exhaust and take the corresponding sequence number as block number for verification.

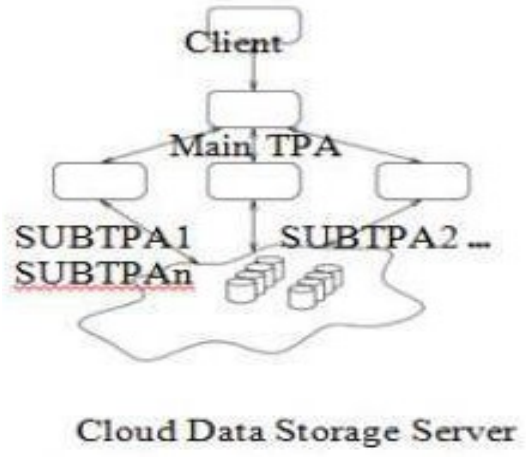

Fig. 1 Block Diagram of Distributed Audit System Architecture

For example, consider the TDK for the SUBTPA1 and SUBTPA2 are 10101 and 01010 respectively. Let, the generated Sobol random sequence is {1216, 5312, 3264, 7360, 704, 4800, 2752, 6848, 1728}, where file blocks are numbered from 0 to 8191. If we place the TDK for SUBTPA1 on the left end of the generated sequence and takes the block numbers corresponding to the 1, after that we slides the string to the right to the same length of the TDK and apply the same procedure then it generates the subtask for SUBTPA1 in (1) and similarly for SUBTPA2 in (2).

1216, 5312, 3264, 7360, 704, 4800, 2752, 6848, 1728}

1 0 1 0 1

*{*1216, 5312, 3264, 7360, 704, 4800, 2752, 6848,1728*}*

1 0 1 0 1

*{*1216, 3264, 704, 4800, 6848*}* (1)

*{*1216, 5312, 3264, 7360, 704, 4800, 2752, 6848, 1728*}*

0 1 0 1 0

*{*1216, 5312, 3264, 7360, 704, 4800, 2752, 6848, 1728*}*

0 1 1 0

(2)

In our protocols, we use two types of TDK for uniformly distribute the task among SUBTPAs and sometime, we adjust the TDK length to balance the subtask for each SUBTPA.

III. THEORETICAL BACKGROUND

A. SOBOL SEQUENCE

Sobol Sequence [3], [4] is a low discrepancy, quasi- random sequences that generates sequences between the interval [0, 1). One salient features of this sequence is that the sequences are Our distributed verification protocols are based on the probabilistic verification scheme and we classify our protocols into two different types depending on the task distribution. First, we are describing our basic protocol based on the simple partition approach. In the second, we use TDK to partition the task. To enhance the performance of our protocols, we used (m, n) threshold scheme [15] with <, where Coordinator can stop the audit operation or detect the fault region after taking responses from any subset of out of SUBTPAs, because each subtask is uniformly distributed over the entire file blocks due to use of Sobol Sequence.

A. Protocol 1: Simple Partition with Threshold Seheme

In the first protocol, the Coordinator randomly chooses one Sobol random key , generate the Sobol Random Block Sequence by using (·), where consist one randomly chosen primitive polynomial, of order out of (2−1)/ primitive polynomials [4], randomly chosen initial values , where ∈ {1, 2, . . . , }, and values respectively. In the next step, partition the generated sequence $\mathcal{L}$ by using partition function

(·), with partition length and denotes each subsequence as , should maintain the equivalence relation property and also maintain the uniformity property. Algorithm 1 gives the detail of key generation and Distribution phase.

ALGORITHM 1: KEY GENERATION & DISTRIBUTION

1 Coordinator randomly chooses one Primitive Polynomial, of degree and initializationnumber
, ∈ {1,2, . . .,};
2 Coordinator decides Sobol Random Keyas
= , , , , ,
3 Generated Sequence
$\mathcal{L} \leftarrow ()$
4 Multiply CONSTANT *powers of 2* with $\mathcal{L}$, to make each element as integer blocknumber;
5 Coordinator Determines the Number of SUBTPAs, , and threshold value,;
6 **for** ←1*to***do**
7 ←(,$\mathcal{L}$)
8 **end**
9 whereis the length of each partition,and
$\mathcal{L} \subseteq$ {1,…, }
= ∈ {,1 , . . . , };
10 DistributesSubtask, to;

uniformly distributed over the interval [0, 1). Also, it maintains uniformity for any segment out of the sequence. That means Sobol Sequence is segment wise uniform. For generating this sequence, we need a primitive polynomial, of degree over the finite field $\mathbb{Z}2$, and direction numbers[3].

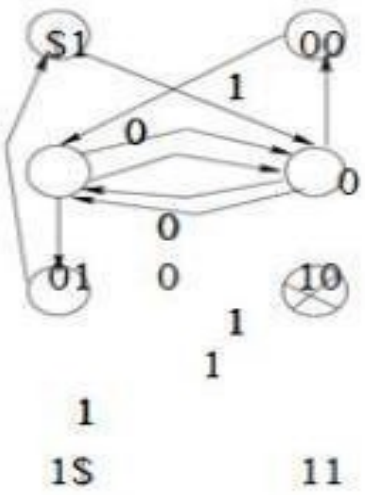

Fig. 2. Modified de Bruijn Graph Corresponding to the string 10010101

Our distributed verification protocols are based on the probabilistic verification scheme and we classify our protocols into two different types depending on the task distribution. First, we are describing our basic protocol based on the simple partition approach. In the second, we use TDK to partition the task. To enhance the performance of our protocols, we used (m, n) threshold scheme [15] with <, where Coordinator can stop the audit operation or detect the fault region after taking responses from any subset of out of SUBTPAs, because each subtask is uniformly distributed over the entire file blocks due to use of Sobol Sequence.

A. Protocol 1: Simple Partition with Threshold Seheme

In the first protocol, the Coordinator randomly chooses one Sobol random key , generate the Sobol Random Block Sequence by using (·), where consist one randomly chosen primitive polynomial, of order out of (2−1)/ primitive

polynomials [4], randomly chosen initial values , where

∈ {1, 2, . . . , }, and values respectively. In the next step, partition the generated sequence $\mathcal{L}$ by using partition function

(·), with partition length and denotes each subsequence as , should maintain the equivalence relation property and also

maintain the uniformity property. Algorithm 1 gives the detail of key generation and Distribution phase.

IV. EFFICIENT DISTRIBUTED ANALYSIS OF PROTOCOL1 VERIFICATION PROTOCOL

Protocol 1 follows the Centrally Controlled and Distributed Data paradigm, where all SUBTPAs are controlled by the Coordinator but communicate to any Cloud Data Storage Server for verification. Here, Coordinator will decide the partition length, , and divides the sequence to each

. Due to the use of Sobol sequence each subsequence must be uniform. After partitioning the sequence, the Coordinator will send the subsequence, , to each .

This protocol gives very good performance to detect errors in the file blocks. Nevertheless, for sending to , from the Coordinator takes extra network band-width. Although, it can not take any extra care about the critical data. To reduce the bandwidth usage and increase the efficiency, and also, taking extra care about critical data, we

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-11, 2017
DOI:10.21276/ijcesr.2017.4.11.22

device the Task Distribution Key (TDK) to divide the sequence to subsequences. Our second scheme describes about TDK based techniques in more details.

ALGORITHM 2:DISRTIBUTED CHALAND PROOF VERIFI CATION( )

```
1                Calculates 10%
2                  of; foreachdo
3                 ← h( ); ←
4                 ⌊(10/100)∗ ⌋;
5                   End
6                for ←1to10 do
7                  for  ←1toanddo
8          h     .[ ] ←[];
9        end
10 Send h , as a challenge to theCloud Server; 11 Wait
for the Proof, , from anyServer;
12                                , ←();
13             if    , equals to Stored Metadatathen
14                              [ ] ←;
15                            Else
16                              [ ] ←;
17                          Send Signal, , ,to the
                              Coordinator;
18       end
19   end
```

B. Protocol 2: TASK DISTRIBUTION KEY BASED DISTRIBUTIONSCHEME

In our second protocol, Coordinator and each will know the Sobol random key, , for generating the Sobol random sequence. In each new verification, Coordinator decides the parameters to generate the Sobol Random Key, and publicly send to all . In addition, Coordinator generates number of random TDKs, , and distributes among SUBTPAs by using String Reconciliation Protocol [1]with some modifications.

each SUBTPA will generate Sobol Random Sequence and interpret their subsequence by using their own TDK. We have given Sobol Random key, TDK generation and distribution in Algorithm 3. Algorithm 4, describes about subtask interpretation, distributed challenge and verification for protocol 2.

In this protocol, we use two types of TDKs, one is Non-Overlapping TDK and another is Overlapping TDK. Overlap-ping TDK will apply when we want to verify critical data. We give the steps for generating Non-Overlapping TDK as follows:

ALGORITHM 3: KEY GENERATION & DISTRIBUTION

```
1  Coordinator randomly chooses one Primitive
   Polynomial, of degree d and initializationnumber
mi, i∈ {1, 2, . . . , d} ;
2 Coordinator decides Sobol RandomKey,
3 Coordinator Determines the Number of SUBTPAs, n, and threshold value,m;
5   Determine number of '1' s, ξ each TDKwillcontain;
6   TDKLen← n*ξ
7 Generates Random Permutation indexfrom
11end
12 end
19 end
21 end
23 TDKLength isacceptable;
24 else
25 TDK Length adjust to the next nearest Primes;
26 end
27 Generated TDKfor SUBTPAare represented as, os1, os2, os3, . . . , osnrespectively, distributes among SUBTPAs by
   using String Reconciliation Protocol.
```

ALGORITHM 4: DISTRIBUTED CHALAND PROOF VERIFICA TION 2( ):

```
1 Each SUBTPAigenerates Sequence
ℒ←for(SeqLen)
2 Multiply CONSTANT powers of 2 with ℒ, t
make each element as integer blocknumber.;
3 Interpret subsequence by using oSias ri,whe
ℒ← ∪

i∈[1,...,

n]

j∈[1,...,

p] ri,j
4 SUBTPAiCalculates 10% of ri,;
5 for each SUBTPAido
6 l←length(ri,j);
7 Counteri← ⌈(10/100) ∗l⌉;
8 end
9 for k← 1 to 10do
10for S←1toCounteriandt⩽ldo
11 Chali,[s]←ri,j[t];
12 end
13 Send ⟨Chali,⟩to SUBTPAi;
14 Wait for the proof, PRi, from any Cloud Se
15PRi,←Rece();
16ifPRi,equalstoStoredMetadatathen
17Re[k]← TRUE;
18 else
19 Re[k] ← FALSE;
20 Send Signal, ⟨Packeti,, FALSE⟩to
the Coordinator;
21end
22end
```

ANALYSIS OF PROTOCOL 2

In TDK generation phase, we take the mask length as coprime to sequence length or prime length, because after applying TDK on ℒ, subsequence, ri, becomes non uniform, and to make it uniform, we use these adjustment. In

algorithm3, Coordinator generates Sobol Random Key and send to the SUBTPAs. In addition, send different TDK, , for each SUBTPAi. In Algorithm 4, SUBTPAi generates the Sobol Random sequence by using key, or and stored in $\mathcal{L}$. Then, each SUBTPAi interpret their task by using corresponding TDK, , and we denoted subtask for SUBTPAias ri,j and. defined as

$\mathcal{L}= \cup$
i∈[1,...,
n] j∈[1,...,
p]ri,j

where
p=
Sequence Length TDK Length+ £
£= Numberof1

sinfirst
(Sequence Length% TDK Length) lengthinTDK

Then, SUBTPAi will calculate 10% of ri, and creates challenge, Chali, and send to the server and waits for the proof, PRi,. After receiving the proof SUBTPA will verify with the stored mata data, and if the proof is correct then store TRUE in its table and if not match then store FALSE and send a signal to the Coordinator for corrupt file blocks. The Coordinator will receive signals from any subset of moutofn SUBTPAs and ensures the fault location or stop the Audit operation. In the final step, Main TPA will give the Audit result to the Client.

Here, we generalize the integrity verification protocol in a distributed manner. Therefore, we can use our protocols on existing RSA based [11] [13] or ECC [10] based protocol to make distributed RSA or ECC protocols. In the next section, we discuss about the performance of our protocols.

V.IMPLEMENTATION AND EXPERIMENTAL RESULTS

It is very natural that audit activities would increase the communication and computational overhead of audit services. To enhance the performance, we used the String Reconciliation Protocol to distribute the TDK ,that reduces the communication overhead between Main TPA and SUBTPAs. For each new verification Coordinator can change the TDK for any SUBTPA and send only the difference part of the multy set element to the SUBTPA. In addition, we used probabilistic verification scheme based on Sobol Sequence that provides not only uniformity for whole sequence but also for each subsequence, so each SUBTPA will independently verify over the whole file blocks. Thus, there is a high probability to detect fault location very efficiently and quickly. Therefore, Sobol sequence provides strong integrity proof for the remotely stored data. Table I shows comparison between two protocols.

TABLE I PERFORMANCE COMPARISON BETWEEN TWO PROPOSED PROTOCOLS

	Protocol 1	**Protocol 2**
Coordinator Controlled		
Privacy Preserving		
Fault Detection		
Coordinator Computation		
Communication Complexity		less

detection probability for Sobol Random Sequence and Pseudo Random Sequence.

We have shown our experimental results in Table II.

TABLE-II
DETECTION PROBABILITY FOR 1% CORRUPTION OUT OF 300000 BLOCKS

Number of samples as Percentage of total samples		Detection Probability
SobolSequence	*PseudorandomSequence*	
10%	20%	0.6
14%	27%	0.7
16%	30%	0.8
21%	37%	0.85
23%	41%	0.9
26%	49%	0.95
30%	54%	0.9999

VI. CONCLUSION

In this paper, we addressed the efficient Distributed Verification protocol based on the Sobol Random Sequence. We have shown that our protocols uniformly distribute the task among SUBTPAs. Most importantly, our

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-11, 2017
DOI:10.21276/ijcesr.2017.4.11.22

protocols can handle failures of SUBTPAs due to its uniform nature and also gives better performance in case of unreliable communication link. Here, we mainly focused on the uniform task distribution among SUBTPAs to detect the erroneous blocks as soon as possible. We used String Reconciliation Protocol to minimize the communication bandwidth between Coordinator and SUBTPA side. In addition, we reduce the workload at the Server side and also reduce the chance of network congestion at the Server side as well as Coordinator side by distributing the task. Thus, our Distributed Verification Protocol increases the efficiency and robustness of data integrity in Cloud Computing. Generate random block numbers by using Sobol Random generator for a given length, then it must be uniform. In addition, if we simply partition the sequence into subsequence and distributes among various SUBTPAs, then each subsequence must be maintain the uniformity. But, when we use TDK then subtask may or may not be uniform. We saw that when the TDK length is powers of 2, then generated subtask does not maintain the uniformity property. Because, Sobol sequence maintain some pattern, if we take 4 consecutive number then we can see that these numbers are from four region over the Sequence, if we divide the full sequence into four region, and for 8, 16, 32,. . . it also hold. When we placed the TDK over the generated Sequence then Subtask contain those numbers whose corresponding TDK bit is 1 and successively applying this TDK togenerate the subsequence. Thus, if the TDK length power of two then for each successive TDK shifting, the chosen block numbers must be very close to each other and form cluster. If, we take TDK length as prime then in each successive shifting the chosen block numbers are spreading over the segment. Therefore, maintains the uniformity for each subtask or subsequence. Now, if the TDKlength is Coprime meangcd(TDKLengt*h*, Sequence Length*h*) = 1Then there is no factor equals to the power of 2, that means for each successive TDK shifting block numbers are spreading over the whole sequence and maintain the uniformity property for each subtask. Therefore, generated subtask must be uniform if the TDK length relatively prime or prime to the sequence length

VII. REFERENCE

[1]. Aaram, Y., Chunhui, S., and Yongdae, K., ―On Protecting Integrity and Confidentiality of Cryptographic File System for Outsourced Storage‖, In proc.of CCSW‘09, Chicago, Illinois,USA November 13, 2009.

[2]. Alexander, H., Bernardo, P., Charalampos, P., and Roberto, T., ―Efficient Integrity Checking of Un trusted Network Storage‖, In Proceedings Of StorageSS‘08, Fairfax, Virginia, October 31, 2008.

[3]. Alexander, S., Christian, C., Asaf, C., Idit, K., Yan, M., and Dani, S., ―Venus: Verification for Un trusted Cloud Storage, In Procedings Of CCSW‘10, Chicago, Illinois, USA October8, 2010.

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

[5]. Anjie P., and Lei W., ―One Publicly Verifiable Secret Sharing Scheme based on LinearCode‖, In Proc. Of 2010 2nd Conference on Environmental Science and InformationApplication Technology, Jul-2010, pp.260- 262.

[6].Apple―ICloud‖Onlineat http://www.apple.com/icloud/what-is.html2010.

[7]. Armbrust, M., Fox, A., Rean, G., Anthony, D. J., Randy, H. K., Andrew K., Gunho L,David, A. P., Ariel, R., Ion, S., and Matei, Z., ―A view of cloud computing‖, Commun.ACM 53, 2010, pp.50–58.

[8]. Armbrust, M., Fox, A., Rean, G., Anthony, D., J.,, Randy., H. K., Andrew K., Gunho L,David, A., P., Ariel, R., Ion, S., and Matei, Z., ―Above the Clouds: A Berkeley View ofCloud Comput-ng,‖ Tech. Rep. UCBEECS-2009, Univ. California, Berkeley, February 28,2009.