IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 4 Issue 8, August 2017
ISSN (Online) 2348 – 7968 | Impact Factor (2016) – 5.264
www.ijiset.com

Publicly Verifiable Inner Product Evalution Over Outsourced Data Streams under Multiple Keys

K Pavithra[1] , S.Amerasan[2] ,
[2]Associate professor
[1, 2] Department of Computer Science and Engineering, PRIST UNIVERSITY, Thanjavur.
pavithrakeerthivasan@gmail.com

Abstract:

Exchanging data streams to an advantage rich cloud server for internal thing evaluation, a basic building frustrate in various standard stream applications (e.g., quantifiable watching), is addressing various associations and individuals. On the other hand, checking the delayed consequence of the remote count accept an essential part in tending to the issue of trust. Since the outsourced data amassing likely starts from different data sources, it is needed for the structure to have the ability to pinpoint the originator of bungles by administering each data source a unique secret key, which requires the inward thing check to be performed under any two social occasions' differing keys. Regardless, the present game plans either depend on upon a lone key supposition or powerful yet practically inefficient totally holomorphic cryptosystems. In this paper, we focus on the all the more troublesome multi-key circumstance where data streams are exchanged by various data sources with unmistakable keys.

Keywords : Exchanging data streams , unique secret key , cryptosystems

Introduction:-

Few years have witnessed the proliferation of streaming data generated by a variety of applications/systems, such as GPS, Internet traffic, asset tracking, wireless sensors, etc. Retaining a local copy of such exponentially-growing volume of data is becoming prohibitive for resource-constrained companies/ organizations, let alone offering efficient and reliable query services on it. Consider a stream-oriented service (e.g., market analysis, weather forecasting and traffic management). The Cloud Computing Interoperability (CCI) is a hot re-search topic and has been addressed by many scientists, architects, groups etc. A lot of different

approaches and possible solutions are published, but there is no accepted standard or model yet. This paper is a survey of the most influential published CCI models and discusses their possibilities and challenges.[1-3]. The accent in this paper is set to analysis of the Software as a Service (SaaS) CCI model based on adapters. The current state of the cloud computing market and the results of recent Cloud Computing (CC) market surveys are also included in our analysis. The presented conclusion addresses the increasing trend in the usage of cloud computing and the lack of visible result to achieve cloud computing interoperability.[4]. Cloud computing systems provide large-scale infrastructures for high-performance computing that are "elastic" since they are able to adapt to user and application needs. Clouds are used through a service-oriented interface that implements the *-as-a-service paradigm to offer Cloud services on demand. This paper discusses Cloud computing models and architectures, their use in parallel and distributed applications, and examines analogies, differences and potential synergies between Cloud computing and multi-agent systems[5-6]. This analysis is lead having in mind the goal of implementing high performance complex systems and intelligent applications by using of Cloud systems and software agents. The convergence of interests between multi-agent systems that need reliable distributed infrastructures and Cloud computing systems that need intelligent software with dynamic, flexible, and autonomous behavior can result in new systems and applications.[6-7]. Cloud computing focuses on delivery of reliable, secure, fault tolerant, sustainable, and scalable infrastructures for hosting internet-based application services. These applications have different composition, configuration, and deployment requirements. Cloud service providers are willing to provide large scaled computing infrastructure at a cheap prices.[8-10].

Existing system:

File Stream upload with Single Key Verification is risky factor. File Data may easily hack or theft by the untrusted Server. The outsourced computation is data cannot be achieved in Single key Algorithm. A probabilistic calculation keep running by every machine takes a tag alone as information, and yields an open key and a mystery key.

Proposed system:

The outsourced computation is data is more secured. Public verification property is banned. The publicly and efficiently verify the inner product evaluation over the outsourced data streams under multiple keys still make more security and accessing data is

efficient. A probabilistic calculation keep running by every machine takes a security parameter as information, and yields an open key and a mystery key. A (perhaps) probabilistic calculation keep running by machine, takes as info its mystery key, the current discrete time and information and yields a freely unquestionable tag.

Methodology:

1. *User Interface Design:*

To connect with server user must give their username and password then only they can able to connect the server. If the user already exits directly can login into the server else user must register their details such as username, password and Email id, into the server. Server will create the account for the entire user to maintain upload and download rate. Name will be set as user id. . Logging in is usually used to enter a specific page.

2. *File Splitting*:

In this module the file being uploaded will be converted from a plain text to cipher text i.e., encrypting the normal file to cipher text file and it will be split into 4 parts with 4 tokens.

3. *Multiple Token Generations:*

In this module the file being split into 4 parts contains 2 bit token for each split file which will be then combine with 6bit key generated by the random Manner for file encryption we used DES(DATA ENCRYPTION STANDARD)

4. *View/Read File:*

For reading each file which have been uploaded and split into 4 parts we should be owner of the file otherwise we should know the four 8bit tokens which have been combined by random algorithm after reading the file you can also download the file otherwise with wrong tokens you can get only cipher text of the content.

5. *Requesting/Response file:*

Requesting a file means as you are not owner of the file but you need to read or download the file that is possible only by owner approval. The person who need the file must be requested to file owner for the tokens once the tokens have been given by the owner the person can able to read/view the file with the token otherwise the person can’t able to get the file.

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 4 Issue 8, August 2017
ISSN (Online) 2348 – 7968 | Impact Factor (2016) – 5.264
www.ijiset.com

Implementation:-

To connect with server user must give their username and password then only they can able to connect the server. If the user already exits directly can login into the server else user must register their details such as username, password and Email id, into the server. Server will create the account for the entire user to maintain upload and download rate. Name will be set as user id. . Logging in is usually used to enter a specific page.

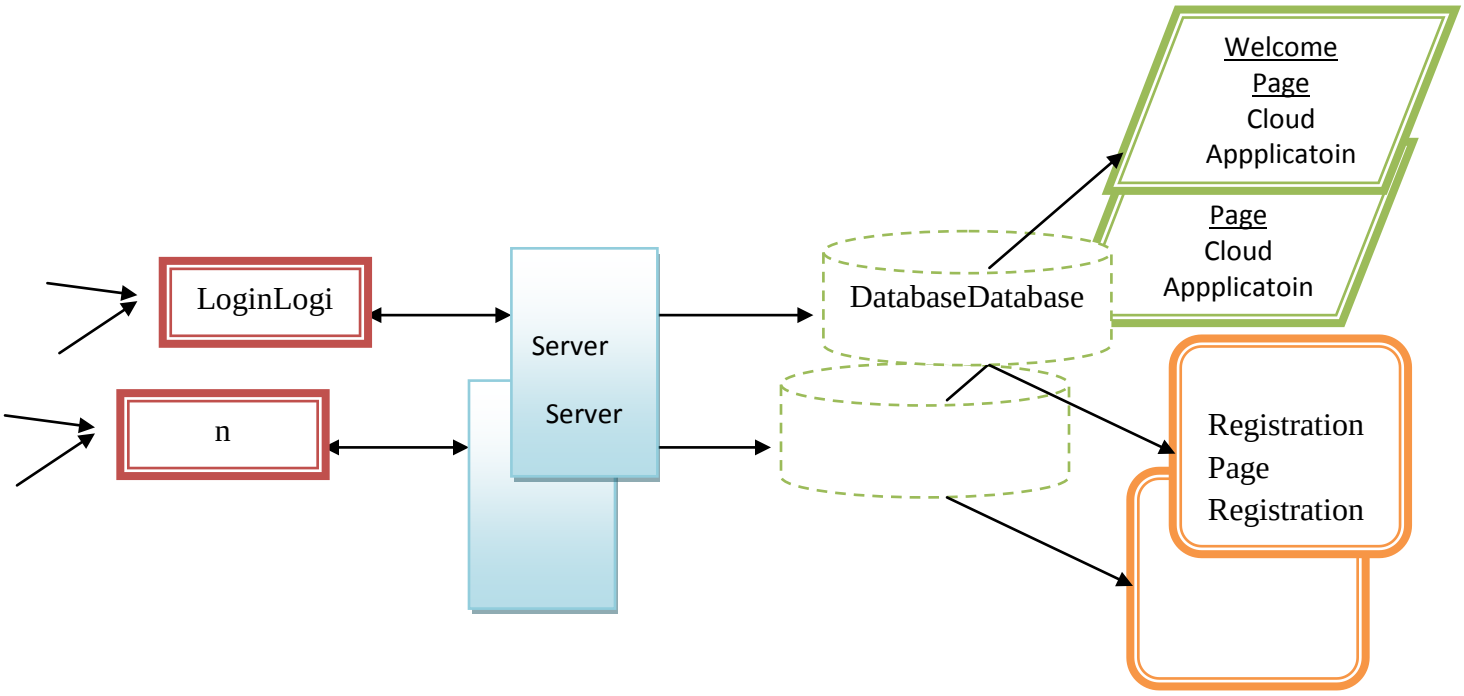

Fig 1 user interface

In this module the file being uploaded will be converted from a plain text to cipher text i.e., encrypting the normal file to cipher text file and it will be split into 4 parts with 4 tokens.

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 4 Issue 8, August 2017
ISSN (Online) 2348 – 7968 | Impact Factor (2016) – 5.264
www.ijiset.com

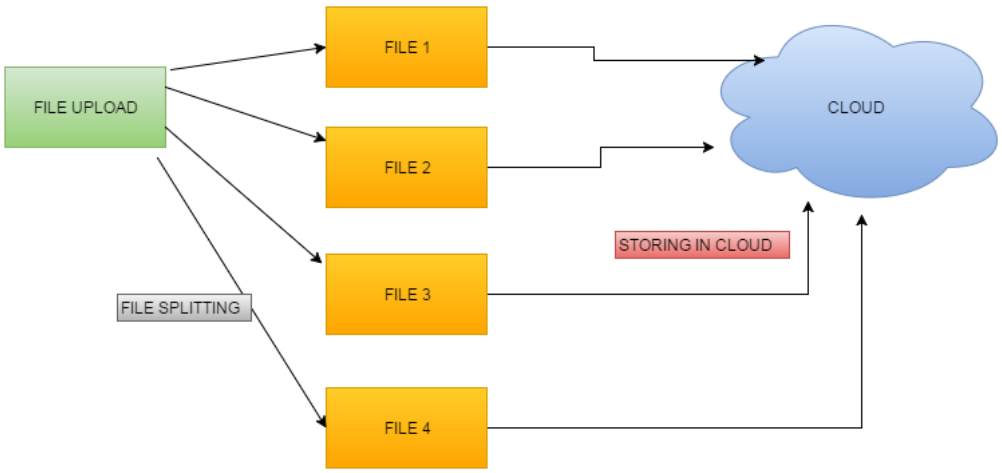

Fig 2 Implementation in File upload

In this module the file being split into 4 parts contains 2 bit token for each split file which will be then combine with 6bit key generated by the random Manner for file encryption we used DES(DATA ENCRYPTION STANDARD)

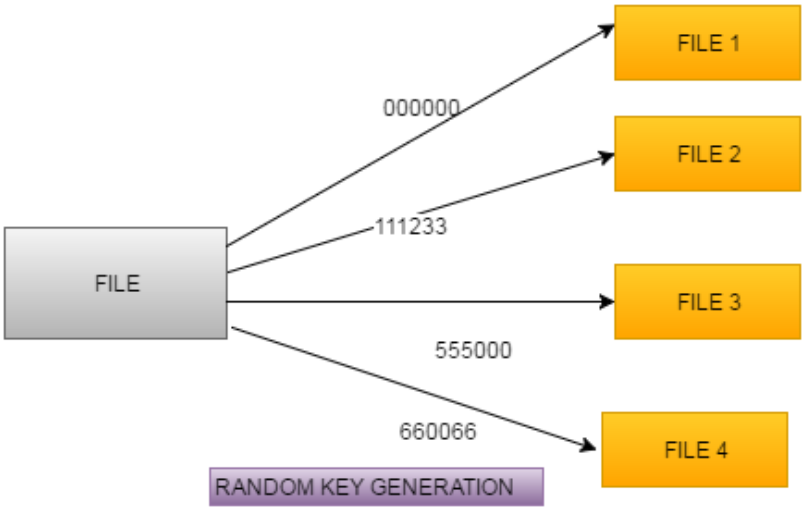

Fig 3 Implementation in Multiple token generation

For reading each file which have been uploaded and split into 4 parts we should be owner of the file otherwise we should know the four 8bit tokens which have been combined by random algorithm after reading the file you can also download the file otherwise with wrong tokens you can get only cipher text of the content.

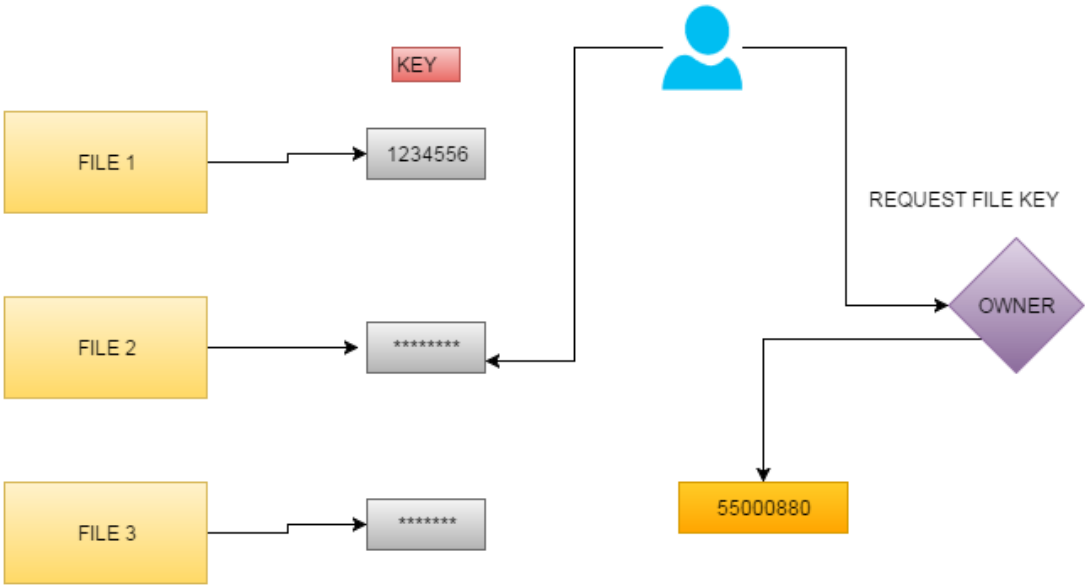

Fig 4 Requesting Response File

Software Testing:-

1. *Unit testing*

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2. *Functional test*

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

3. *System Test*

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

4. *Performance Test*

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

5. *Integration Testing*

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

6. *Acceptance Testing*

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updation process

7. *Build the test plan*

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

Conclusions

In this paper, we introduce a novel holomorphic verifiable tag technique, and design an efficient and publicly verifiable inner product computation scheme on the dynamic outsourced

data streams under multiple keys. We also extend the inner product scheme to support matrix product. Compared with the existing works under the single-key setting, our scheme aims at the more challenging *multi-key* scenario, i.e., it allows multiple data sources with different secret key sto upload their *endless data streams* and delegate the corresponding computations to a third party server, while the traceability can still be provided on demand. Furthermore, any *keyless* client is able to *publicly* verify the validity of the returned computation result. Security analysis shows that our scheme is provable secure under the CDH assumption in the random oracle model. Experimental results demonstrate that our protocol is practically efficient in terms of both communication and computation cost.

References

[1] The NIST Definition of Cloud Computing. National Institute ofStandards and Technology, Commerce.http://www.csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[2] M. Armbrust et al. Above the clouds: A Berkeley view of cloudcomputing. Tech. Rep. UCB/EECS-2009-28, EECS Department,

U.C. Berkeley, Feb 2009.[3] D. Bernstein et al. Blueprint for the Intercloud - Protocols andFormats for Cloud Computing Interoperability. Proc. 4th Int.Conf. Internet and Web Applications and Services pp. 328–336.,Venice, May 2009.

[4] M. Wooldridge, An Introduction to Multiagent Systems, JohnWiley & Sons, 2002.

[5] K. M. Sim. “Agent-based Cloud Computing,” IEEE Transactionson Services Computing. vol. 5, no. 4, Oct.-Dec., 2012, pp. 564-577.

[6] K. M. Sim, Complex and concurrent negotiations for multipleinterrelated e-markets, IEEE Trans. Cybernet. 43 (1), pp. 230–245,

2013.

[7] S. Son and K. M. Sim, A price-timeslot negotiation for cloudservice reservation, IEEE Trans. Syst. Man Cybernet. B 42 (3), pp.

713–728, 2012.

[8] J. O. Gutierrez-Garcia and K. M. Sim, Agent-based cloud servicecomposition, Appl. Intell. 38 (3), pp. 436–464, 2013.

[9] J. O. Gutierrez-Garcia and K. M. Sim, Family of heuristics foragent-based elastic cloud bag-of-tasks concurrent scheduling,

Future Generat. Comput. Syst. 29 (7), pp. 1682–1699, 2013.

[10] J. O. Gutierrez-Garcia and K. M. Sim, Agent-based cloudworkflow execution, Integr. Comput. Aided Eng. 19 (1), pp.
39–56, 2012.