International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 6 Issue 6|| June 2017 || PP. 14-19

Extensive Security and Performance Analysis Shows the Proposed Schemes Are Provably Secure and Highly Efficient

Mr. Kandunuri Ramakrishna[1], Mr. Rokesh Kumar Y[1], Mr. U.Rakesh[2]
[1]*Ph.D Research Scholar, Sri Satya Sai University of Technology & Medical Sciences (SSSUTMS), Sehore, Madhya Pradesh, India.*
[2]*Assistant Professor, Department of Computer Science Engineering and Technology SV Engineering college for women, Andhra Pradesh, India*

Abstract*: In this paper, we utilize the public key based homomorphism authenticator and uniquely integrate it with random mask technique to achieve a privacy-preserving public auditing system for cloud data storage security while keeping all above requirements in mind. To support efficient handling of multiple auditing tasks, we further explore the technique of bilinear aggregate signature to extend our main result into a multi-user setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis shows the proposed schemes are provably secure and highly efficient. We also show how to extent our main scheme to support batch auditing for TPA upon delegations from multi-users.*

keywords*: network of nodes, Data Sanitization, Batch Auditing, privacy-preserving ,Denial of Service, Bayesian Belief Networks, data computation, Error Estimation, test defect data, Software metrics, empirical data..*

I. INTRODUCTION

Cloud computing is the long dreamed vision of computing as a utility, where users can remotely store their data into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources. By data outsourcing, users can be relieved from the burden of local data storage and maintenance. Thus, enabling public audit ability for cloud data storage security is of critical importance so that users can resort to an external audit party to check the integrity of outsourced data when needed. To securely introduce an effective third party auditor (TPA), the following two fundamental requirements have to be met: TPA should be able to efficiently audit the cloud data storage without demanding the local copy of data, and introduce no additional on-line burden to the cloud user. Specifically, our contribution in this work can be summarized as the following three aspects: We motivate the public auditing system of data storage security in Cloud Computing and provide a privacy-preserving auditing protocol, i.e., our scheme supports an external auditor to audit user's outsourced data in the cloud without learning knowledge on the data content. To the best of our knowledge, our scheme is the first to support scalable and efficient public auditing in the Cloud Computing. In particular, our scheme achieves batch auditing where multiple delegated auditing tasks from different users can be performed simultaneously by the TPA. We prove the security and justify the performance of our proposed schemes through concrete experiments and comparisons with the state-of-the-art.

II. SYSTEM OVERVIEW

To support efficient handling of multiple auditing tasks, we further explore the technique of bilinear aggregate signature to extend our main result into a multi-user setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis shows the proposed schemes are provably secure and highly efficient. We also show how to extent our main scheme to support batch auditing for TPA upon delegations from multi-users.

2.1 Modules Explanation

1. Privacy-Preserving Public Auditing Module
2. Batch Auditing Module
3. Data Dynamics Module

2.1.1 Privacy-Preserving Public Auditing Module: Homomorphic authenticators are unforgivable verification metadata generated from individual data blocks, which can be securely aggregated in such a way to assure an auditor that a linear combination of data blocks is correctly computed by verifying only the aggregated authenticator. Overview to achieve privacy-preserving public auditing, we propose to uniquely integrate the homomorphic authenticator with random mask technique. In our protocol, the linear combination of sampled blocks in the server's response is masked with randomness generated by a pseudo random function (PRF). The proposed scheme is as follows: Setup Phase, Audit Phase

2.1.2 Batch Auditing Module :

- There are K users having K files on the same cloud
- They have the same TPA
- Then, the TPA can combine their queries and save in computation time
- The comparison function that compares the aggregate authenticators has a property that allows checking multiple messages in one equation
- Instead of 2K operation, K+1 are possible

2.1.3 Data Dynamics Module :

- The data on the cloud may change according to applications
- This is achieved by using the data structure Merkle Hash Tree (MHT)
- With MHT, data changes in a certain way; new data is added in some places
- There is more overhead involved ; user sends the tree root to TPA
- This scheme is not evaluated in the paper

–

III. THE WORKING PRINCIPLE

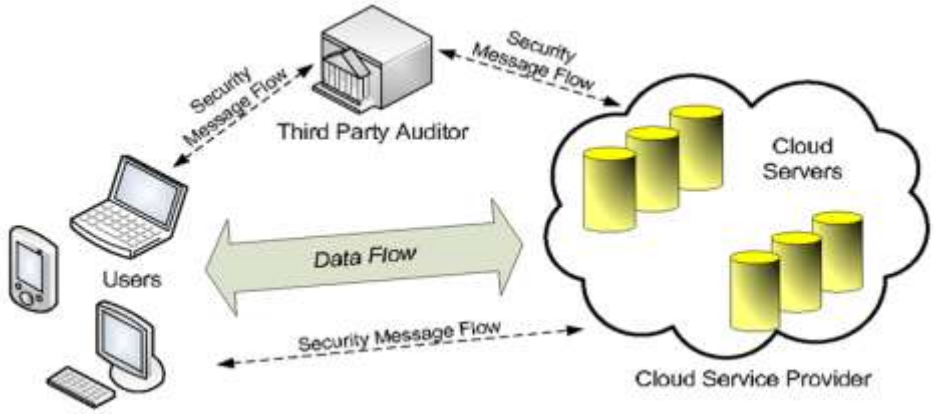
To enable privacy-preserving public auditing for cloud data storage under the aforementioned model, our protocol design should achieve the following security and performance guarantee:

1) Public audit ability: to allow TPA to verify the correctness of the cloud data on demand without retrieving a copy of the whole data or introducing additional on-line burden to the cloud users.

Regulatory/Compliance Requirements :
The database will have a functional audit trail
The system will limit access to authorized users
The spreadsheet can secure data with electronic signatures

Security Requirements
Member of the Data Entry group can enter requests but not approve or delete requests
Members of the Managers group can enter or approve a request, but not delete requests
Members of the Administrators group cannot enter or approve requests, but can delete requests
The functional specification describes what the system must do; how the system does it is described in the Design Specification.
If a User Requirement Specification was written, all requirements outlined in the user requirement specification should be addressed in the functional requirements.

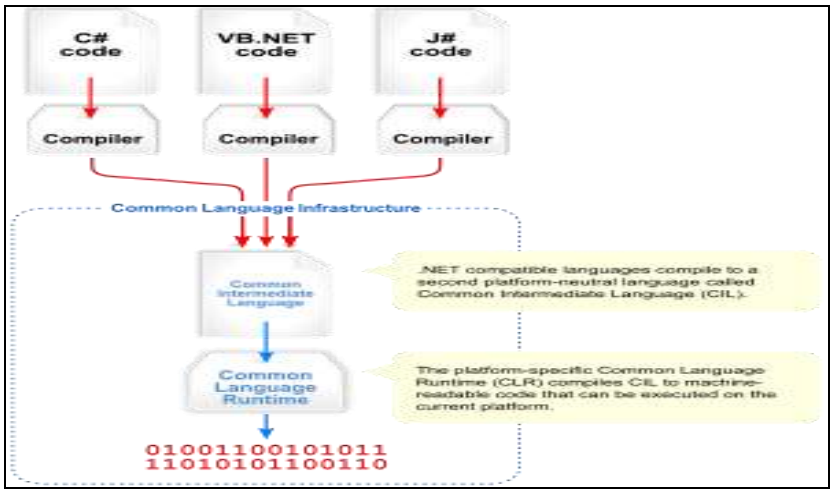
2) Storage correctness: to ensure that there exists no cheating cloud server that can pass the audit from TPA without indeed storing users' data intact.
3) Privacy-preserving: to ensure that there exists no way for TPA to derive users' data content from the information collected during the auditing process.
4) Batch auditing: to enable TPA with secure and efficient auditing capability to cope with multiple auditing delegations from possibly large number of different users simultaneously.
5) Lightweight: to allow TPA to perform auditing with minimum communication and computation overhead



3.1 Functional Requirements

- Functional Requirements refer to very important system requirements in a software engineering process (or at micro level, a sub part of requirement engineering) such as technical specifications, system design parameters and guidelines, data manipulation, data processing and calculation modules etc.

- Functional Requirements are in contrast to other software design requirements referred to as Non-Functional Requirements which are primarily based on parameters of system performance, software quality attributes, reliability and security, cost, constraints in design/implementation etc.
- The key goal of determining "functional requirements" in a software product design and implementation is to capture the required behavior of a software system in terms of functionality and the technology implementation of the business processes.
- The Functional Requirement document (also called Functional Specifications or Functional Requirement Specifications), defines the capabilities and functions that a System must be able to perform successfully.
- Functional Requirements should include:
- Descriptions of data to be entered into the system
- Descriptions of operations performed by each screen
- Descriptions of work-flows performed by the system
- Descriptions of system reports or other outputs
- Who can enter the data into the system?
- How the system meets applicable regulatory requirements
- The functional specification is designed to be read by a general audience. Readers should understand the system, but no particular technical knowledge should be required to understand the document.
- Functional requirements should include functions performed by specific screens, outlines of work-flows performed by the system and other business or compliance requirements the system must meet.
- Interface requirements
- Field accepts numeric data entry
- Field only accepts dates before the current date
- Screen can print on-screen data to the printer
- Business Requirements
- Data must be entered before a request can approved
- Clicking the Approve Button moves the request to the Approval Workflow
- All personnel using the system will be trained according to internal training strategies
- Regulatory/Compliance Requirements
- The database will have a functional audit trail
- The system will limit access to authorized users
- The spreadsheet can secure data with electronic signatures
- Security Requirements
- Member of the Data Entry group can enter requests but not approve or delete requests
- Members of the Managers group can enter or approve a request, but not delete requests
- Members of the Administrators group cannot enter or approve requests, but can delete requests
- The functional specification describes what the system must do; how the system does it is described in the Design Specification.
- If a User Requirement Specification was written, all requirements outlined in the user requirement specification should be addressed in the functional requirements.



3.2 Non Functional Requirements

- All the other requirements which do not form a part of the above specification are categorized as Non-Functional Requirements.
- A system may be required to present the user with a display of the number of records in a database. This is a functional requirement.
- How up-to-date this number needs to be is a non-functional requirement. If the number needs to be updated in real time, the system architects must ensure that the system is capable of updating the displayed record count within an acceptably short interval of the number of records changing.
- Sufficient network bandwidth may also be a non-functional requirement of a system.

–

IV. IMPLEMENTATION OF SYSTEM

The design of the .NET Framework allows it to theoretically be platform agnostic, and thus cross-platform compatible. That is, a program written to use the framework should run without change on any type of system for which the framework is implemented. Microsoft's commercial implementations of the framework cover Windows, Windows CE, and the Xbox 360. In addition, Microsoft submits the specifications for the Common Language Infrastructure (which includes the core class libraries, Common Type System, and the Common Intermediate Language), the C# language, and the C++/CLI language to both ECMA and the ISO, making them available as open standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

The core aspects of the .NET framework lie within the Common Language Infrastructure, or CLI. The purpose of the CLI is to provide a language-neutral platform for application development and execution, including functions for exception handling, garbage collection, security, and interoperability. Microsoft's implementation of the CLI is called the Common Language Runtime or CLR.

Assemblies

The intermediate CIL code is housed in .NET assemblies. As mandated by specification, assemblies are stored in the Portable Executable (PE) format, common on the Windows platform for all DLL and EXE files. The assembly consists of one or more files, one of which must contain the manifest, which has the metadata for the assembly. The complete name of an assembly (not to be confused with the filename on disk) contains its simple text name, version number, culture, and public key token. The public key token is a unique hash generated when the assembly is compiled, thus two assemblies with the same public key token are guaranteed to be identical from the point of view of the framework. A private key can also be specified known only to the creator of the assembly and can be used for strong naming and to guarantee that the assembly is from the same author when a new version of the assembly is compiled (required to add an assembly to the Global Assembly Cache).

Metadata

All CLI is self-describing through .NET metadata. The CLR checks the metadata to ensure that the correct method is called. Metadata is usually generated by language compilers but developers can create their own metadata through custom attributes. Metadata contains information about the assembly, and is also used to implement the reflective programming capabilities of .NET Framework.

Security:.NET has its own security mechanism with two general features: Code Access Security (CAS), and validation and verification. Code Access Security is based on evidence that is associated with a specific assembly. Typically the evidence is the source of the assembly (whether it is installed on the local machine or has been downloaded from the intranet or Internet). Code Access Security uses evidence to determine the permissions granted to the code. Other code can demand that calling code is granted a specified permission. The demand causes the CLR to perform a call stack walk: every assembly of each method in the call stack is checked for the required permission; if any assembly is not granted the permission a security exception is thrown.

When an assembly is loaded the CLR performs various tests. Two such tests are validation and verification. During validation the CLR checks that the assembly contains valid metadata and CIL, and whether the internal tables are correct. Verification is not so exact. The verification mechanism checks to see if the code does anything that is 'unsafe'. The algorithm used is quite conservative; hence occasionally code that is 'safe' does not pass. Unsafe code will only be executed if the assembly has the 'skip verification' permission, which generally means code that is installed on the local machine.

4.1. SYSTEM DESIGN: The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.A UML system is represented using five different views that describe the system from distinctly different perspective.

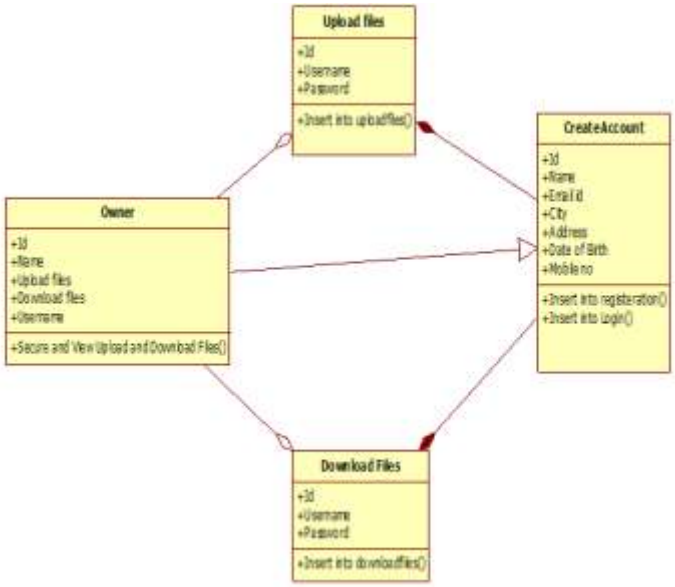


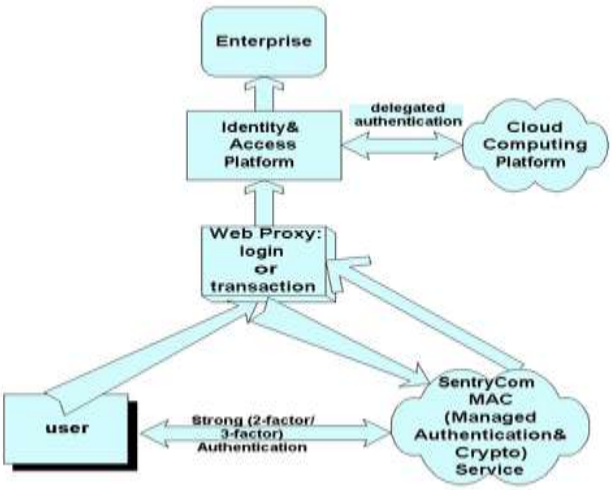
Fig 2: Class Diagram

4.2 Algorithm:

A public auditing scheme consists of four algorithms (KeyGen, SigGen, GenProof, VerifyProof).

- KeyGen: key generation algorithm that is run by the user to setup the scheme
- SigGen: used by the user to generate verification metadata, which may consist of MAC, signatures or other information used for auditing
- GenProof: run by the cloud server to generate a proof of data storage correctness
- Verify Proof: run by the TPA to audit the proof from the cloud server

4.3. Flow Chart:



V. CONCLUSION

We proposed a lightweight authentication protocol based on Gen2 to resist various attacks. The proposed tag uses no cryptographic function, and hence, is suitable for low-cost RFIDs. Without changing the protocol flow of Gen2, the existing reader can read both Gen2 tags and Gen2þ tags. Gen2þ provides sufficient security level for real-world settings. We analyzed the number of rounds required and the period of key update for practical deployment. To support efficient handling of multiple auditing tasks, we further explore the technique of bilinear aggregate signature to extend our main result into a multi-user setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis shows the proposed schemes are provably secure and highly efficient. We also show how to extent our main scheme to support batch auditing for TPA upon delegations from multi-users.

REFERENCES

[1] 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, March2010.

[2] D. Boneh, B. Lynn, and H. Shacham, "Short signatures fromthe Weil pairing," J. Cryptology, vol. 17, no. 4, pp. 297–319,2004.

[3] A. L. Ferrara, M. Greeny, S. Hohenberger, and M. Pedersen,"Practical short signature batch verification," in Proceedings of CT-RSA, volume 5473 of LNCS. Springer-Verlag, 2009, pp. 09–324.

[4] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik,"Scalable and efficient provable data possession," in Proc. ofSecureComm'08, 2008, pp. 1–10.

[5] C.Wang, Q.Wang, K. Ren, andW. Lou, "Ensuring data storage security in cloudcomputing," in Proc. of IWQoS'09, July 2009,pp. 1–9.

[6] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia,"Dynamic provable data possession," in Proc. of CCS'09, 2009,pp. 213–222.

[7] R. C.Merkle, "Protocols for public key cryptosystems," in Proc.of IEEE Symposium on Security and Privacy, Los Alamitos, CA,USA, 1980.

[8] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia,"Dynamic provable data possession," in Proc. of CCS'09, 2009,pp. 213–222.

[9] R. C. Merkle, "Protocols for public key cryptosystems," in Proc.of IEEE Symposium on Security and Privacy, Los Alamitos, CA,USA, 1980.

[10] G. Ateniese, S. Kamara, and J. Katz, "Proofs of storage from homomorphic identification protocols," in ASIACRYPT, 2009, pp. 319–333.

[11] M. Bellare and G. Neven, "Multi-signatures in the plain publickey model and a general forking lemma," in ACM Conference on Computer and Communications Security, 2006, pp. 390–399.

[12] Y. Dodis, S. P. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in TCC, 2009, pp. 109–127.

Biography:

Mr.K.Ramakrishna,(Ph.D) Research Scholar in Sri Satya Sai University of Technology and Medical Sciences, Sehore (M.P) India. He received the master of technology degree in VNR Vignana Jyothi Institute of Engineering and Technology- Jawaharlal Nehru Technological University Hyderabad, India . He received the bachelor of technology degree in The Vazir Sultan College of Engineering And technology, kakatiya university ,Warangal, India.His Research Interests Include mobile ad-hoc networks , Data Mining, Information Security, Software Testing, Software Engineering, mobile communication and cloud computing.

Mr. Y. Rokesh kumar, Ph.D Research Scholar in Sri Satya Sai University of Technology and Medical Sciences, Sehore (M.P.) India. He Has 5+ Years Teaching Experience, His Research Interests Include cloud computing , Data Mining, Information Security, Software Testing, mobile communication and mobile ad-hoc networks.

Mr.U.Rakesh, presently working as an assistant professor in computer science engineering and technology department, SV Engineering college for women's , India. He received the master of technology degree in VNR Vignana Jyothi Institute of Engineering and Technology- Jawaharlal Nehru Technological University Hyderabad, India . He received the bachelor of technology degree in Jawaharlal Nehru Technological University Hyderabad, India. He Has 6+ Years Teaching Experience, His Research Interests Include mobile ad-hoc networks , Data Mining, Information Security, Software Testing, mobile communication and cloud computing.
