International Journal of Enhanced Research in Management & Computer Applications
ISSN: 2319-7471, Vol. 6 Issue 6, June-2017, Impact Factor: 3.578

Secure Protocols for Developed Cloud Computing using Trusted Platform Module (TPM)

Adapa Gopi [1], Dr. Rashmi Agarwal[2]

[1]Ph.D. Scholar, Department of Computer Science & Engineering, Madhav University
[2]Associate Professor, Department of Computer Science & Engineering, Madhav University

ABSTRACT

Cloud Computing is a term used to describe both a platform and type of application. Cloud Computing differs from traditional computing paradigms as it is scalable, can be encapsulated as an abstract entity which provides different level of services to the clients, driven by economies of scale and the services are dynamically configurable. Data stored in third party storage systems like the cloud might not be secure since confidentiality and integrity of data are not guaranteed. Though cloud computing provides cost-effective storage services. Hence, many organizations and users may not be willing to use the cloud services to store their data in the cloud until certain security guarantees are made. In this paper, a solution to the problem of securely storing the client's data by maintaining the confidentiality and integrity of the data within the cloud is developed. The proposed protocols are developed which ensure that the client's data is stored only on trusted storage servers, replicated only on trusted storage servers, and guarantee that the data owners and other privileged users of that data access the data securely.

Keywords- Cloud Computing, Trusted Storage, and Security.

1. INTRODUCTION

As a platform it supplies, configures and reconfigures servers, while the servers can be physical machines or virtual machines. On the other hand, Cloud Computing describes applications that are extended to be accessible through the internet and for this purpose large data centers and powerful servers are used to host the web applications and web services. Unencrypted data of the client cannot be stored in the cloud because the cloud provider will have access to the data and hence the confidentiality of the data will be lost. The encryption and decryption of files is transparent to the user and the application. The encrypted key is decrypted with the user's private key. This is because the encryption keys used to encrypt data are stored in the disk and even when the encryption keys are encrypted, they are encrypted using the public key of the user of that storage node. The user of the storage node in his cloud is the system administrator who has the maximum privileges on that storage server. Thus, he could easily get hold of the encryption/decryption keys of the encrypted file system stored in the disk and thereby decrypt the client's data and can even modify it. Hence the confidentiality and integrity of the client's data might be lost. Loss of data confidentiality and integrity is undesirable for a client. They are the two main security issues for a client who is using the cloud services to store his data. To enable a client to establish trust in the cloud provider's ability to securely store his data within the cloud, we need a solution to achieve confidentiality and integrity of client's data within the cloud [1] and [3] and [4]. In this paper, a solution to the problem of securely storing the client's data by maintaining the confidentiality and integrity of the data within the cloud is developed. The proposed system is called 'A Trusted Storage System for Cloud'.

BACKGROUND

CLOUD ARCHITECTURE

The user interacts with the front-end interface from which he selects a service, for example, to store his files, to access a document, or to run an application. The user's request is transferred to the System Management which finds the appropriate resources to be assigned, and calls upon the Provisioning Services mechanism to provision the resources to the user. The Provisioning Services mechanism contacts the cloud servers and processes the user's request. After

processing the user's request, the cloud system monitor tracks the usage of resources by the user and records it in his profile. Thus, the cloud provider charges the user according to his cloud usage. All of these management tasks are automated in the cloud computing system [2].

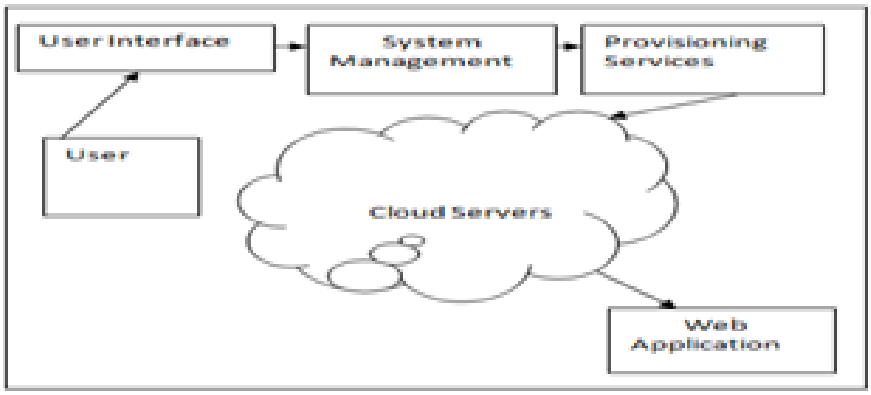


Figure1: Architecture of a Cloud Computing System

CLOUD STORAGE

The user's data is stored in multiple servers of the cloud rather than on a dedicated server like in the traditional data center storage. To the user, it appears as if his data is stored in a particular location but in fact, the data might move from one location to another in time.

The cloud consists of several storage servers and a front-end server/node manager which manages the storage servers within the cloud. A client communicates with the cloud through the front-end server and vice-versa. The client uses a web-based interface to communicate with the front-end server of the cloud. The client sends his data/files via the internet to the cloud for storage. The front-end server after receiving the client's files chooses a storage server within the cloud and sends the client's files to it. The Client's data is replicated within the cloud storage servers for reliability. A storage server sends copies of its data to other storage servers of the cloud [1] and [3].

RELATED WORKS

He and Xu propose a scheme to protect data on a personal computer platform. They use the trusted computing platform developed by the trusted computing group (TCG) to develop a secure and reliable model for user authentication and data encryption. Their model uses a storage protocol to encrypt data and uses Trusted Platform Module (TPM) to authenticate different users of the PC. The host computer has the trusted computing platform installed within itself, i.e., it has TPM installed in it. First, a trusted connection is established between the host computer and the storage device, i.e., the host and the storage device authenticate each other before any data storage takes place within the storage device and before data is accessed from the storage device by the host. Then, security control is provided by storing access control policies within the storage device. The access control policies apply to users, devices and applications. Message control between the host and the storage device is provided by implementing session-oriented secure data transmission [4-6].

Sailor, Doorn and Ward described a trusted computing platform which extends the hardware-rooted trust guarantees of TCG (Trusted Computing Group) technologies to the operating system and all its applications and allows remote parties to check the trust guarantees. The TCG defines standards for measuring and reporting integrity metrics at the system boot time. However, the TCG does not provide information on the trustworthiness of the runtime of the operating system. Sailor et al. provide a solution to this problem. They give a procedure that explains how a challenging party can attest to the software stack of an untrusted machine. Trust is established between two parties after mutual attestation takes place. The kernel of the attested system (system that needs to be attested) is instrumented to generate measurements of postboot events which affect the run-time of the system. The components of the software stack that have semantic value are measured. The measured components are kernel modules, executables and shared libraries, configuration files and other important input files that affect trust into the run-time software stack. The measurements are done as soon as executable content is loaded into the system and before it is executed. The attested system creates and stores a measurement list which is a list of hashes of the software stack components. This list is also stored in the TPM (Trusted Platform Module) for achieving integrity. The measured list and the list stored in the TPM are sent to the challenging party. The challenging party compares the two of them and trusts the measurement list of the attested system if both turn out to be the same. After that, the challenging party compares the received measurement list

with its stored expected list of measurements. If the two turn out to be the same, the challenging party attests to the system and hence the challenging party is assured that the system is running the expected software stack [7] and [8].

PROPOSED TECHNIQUES

We consider the confidentiality and integrity issues of client's data within the cloud and provide a solution to these issues by proposing a framework/system called a Trusted Storage System for Cloud. The system provides data security against the cloud provider, especially against the system administrator who has the maximum number of privileges over the storage nodes in the cloud.

SYSTEM ARCHITECTURE

The proposed system consists of the following entities:

- User/Client
- Trusted Third Party Node (TTPN)
- Front-End Server (FES)
- Group of Storage Servers/Nodes

JOINING OF A STORAGE NODE IN THE TRUSTED STORAGE SYSTEM OF THE CLOUD

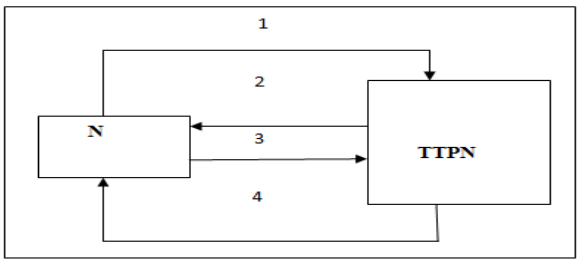


Figure 2: Joining of a storage node in the Trusted Storage System of the Cloud.

A storage node in the cloud must join the Trusted Storage System before it can store a client's data. The TTPN tests if a TPM is being installed on the storage node and if installed, it checks for the correctness of the platform of the storage node. This process is called attestation. After successful attestation of the platform of the storage node by the TTPN, the storage node is deemed trusted and can store a client's data securely.

NOTATIONS

nTTPN: Nonce of the TTPN PCRVN: The PCR value stored in the TPM of node N (Hash of the kernel of the node and sequence of hashes of the software involved in the boot sequence of the node bootstrap loader, BIOS).

MLN: Measurement list of the node N
pri(AIK): Private attestation identity key of the TPM of node N
pub(AIK): Public attestation identity key of the TPM of node N.
C(AIK): Attestation identity key certificate of the TPM of node N.
pub(N): Public key of the node N.
Nid: ID of the node N.
pub(TTPN): Public key of the TTPN.
pri(TTPN): Private key of the TTPN.

Protocol 1
Message 1: {"Join System", Nid}
Message 2: {"Send platform state", nTTPN} pri(TTPN)
Message 3: {{PCRVN ,nTTPN} pri(AIK) , MLN, pub(N), pub(AIK), C(AIK)} pub(TTPN)
Message 4: {"Joined"}pri(TTPN)

Data Storage in the Trusted Storage System

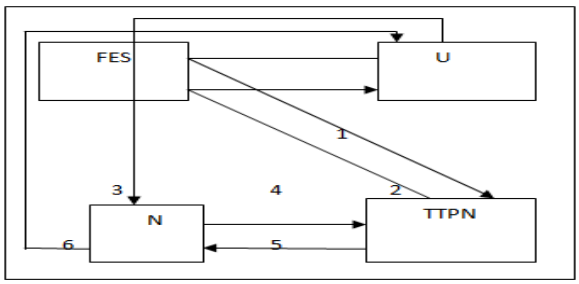


Figure 3: Data Storage in the Trusted Storage System.

NOTATIONS
UK: User Key
Uid: User ID
nU: Nonce of the user
nTTPN: Nonce of the TTPN
nN: Nonce of the node
SK: One time session key
pub(TTPN): Public key of the TTPN
pri(TTPN): Private key of the TTPN
pub(N): Public key of the node
pri(N): Private key of the node
{files}SK: User's/client's files encrypted with the session key
Nid: Node ID

Protocol 2
Message 1: {"Data Send Req", UK ,Uid, nU } pub(TTPN)
Message 2: {{ SK, nU, nTTPN }UK, {H(nU)}pri(TTPN)}
Message 3: {{files}SK, {H({files} SK), nTTPN} pub(TTPN)}
Message 4: {Message 3, (H(Message 3))pri(N), {nN, Nid} pub(TTPN)}
Message 5: {{files}SK, {{nN, UK , Uid, SK, H({files}SK) } pub(N)} pri(TTPN)}
Message 6: {"Data Stored"}UK

The client/user sends a "data send request" (message 1) to the front-end server (FES) of the cloud to store his data in the cloud. The front-end server forwards the message to the TTPN. The user generates a symmetric encryption key, UK, called user key, to use it in further communications with the cloud. Message 1 includes the user key, user ID, Uid, and the user nonce, nU, used for preventing replay attacks. The message is encrypted with the public key of the TTPN so that only the TTPN can decrypt and view the message.

DATA INTEGRITY

During data storage, the EFS computes a hash of the user's data and stores it in the TPM. When the data is accessed by the user, the EFS computes the hash of the user's data and compares it with the hash stored in the TPM. If both the hashes are equal, the EFS is assured that the user's data is not modified. If the hashes are not the same, an integrity breach is identified.

User's data is transmitted securely and is stored only on trusted storage nodes:
The user encrypts his data with the one-time symmetric key, SK, which is created and provided to the user by the TTPN which is trusted by the user. The user sends the encrypted data to the cloud. A storage node within the cloud can decrypt the user's data, re-encrypt it with the EFS keys and store the user's data within it only after getting itself authenticated by the TTPN.

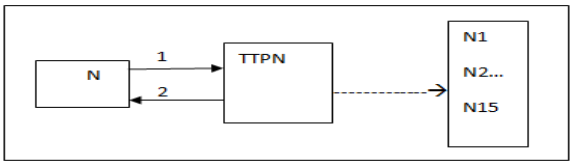
The storage node sends an authentication request to the TTPN by signing the request with its private key. The TTPN authenticates the storage node if it is a member of the trusted storage system by checking its database if it has the public key of the storage node. The TTPN does not authenticate the storage node in the following situations:

DATA REPLICATION

Such storage server replicates its data onto other servers in the cloud to achieve data reliability. The data stored in a trusted storage server should be replicated onto other trusted storage servers only. That is, the data should be replicated onto storage servers which are members of the trusted storage system.

Storage node ready to accept data for replication

In order to ensure that the data for replication is transferred to trusted storage servers, the TTPN maintains a list of trusted storage servers that are ready to accept data for replication. This list is called "ready node list". The ready node list is a dynamic list which keeps changing when new nodes are added to it or existing nodes get deleted from it. A storage server which is ready to accept data for replication sends a "ready for replication" message to the TTPN. The TTPN checks if that server is a member of the trusted storage system. If it is a member, it adds that server to the ready node list.



1. If the storage node is not a member of the trusted storage system, i.e., if the public key of the storage node is not found in the database of the TTPN.
Or
2. If the public key found in the database does not decrypt the signature of the node. This is the case when the storage node is compromised/ rebooted.

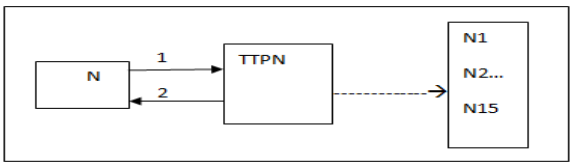


Figure 4: Storage node ready to accept data for replication.

NOTATIONS
pri(N): Private key of the node N
Nid: ID of the node N

Protocol 3
Message 1: {"Ready for replication", {"add to ready node list"} pri(N), Nid}
Message 2: {"Added/ Rejected"}

A storage server which is ready to accept data for replication sends "ready for replication" message which includes its ID and "add to ready node list" message which is encrypted with the private key of the node, pri(N), so that the TTPN can authenticate the node by decrypting it using the public key of the node, pub(N).

When the TTPN receives message 1, checks its database if it has the public key, pub(N), of that node stored, i.e., it checks if that node is a member ("trusted node") of the Trusted Storage System. If the TTPN finds the public key, it decrypts the "add to ready node list" message and thus successfully authenticates and adds the storage node to its ready node list. If the TPPN does not find the public key of that node or if it is not able to decrypt the message with the existing public key of that node, it identifies that the node is not a member of the trusted storage system or that the node has been rebooted and compromised. The TTPN will not add that node to its ready node list. So, the ready node list contains trusted storage nodes only. In response to message 1, the TTPN sends message 2 containing "added/rejected" message. When a node cannot accept any more data for replication, it sends "delete from ready node list" message to the TTPN. The TTPN deletes that node from the ready node list. Ni

Storage node sends data for replication

A storage node N in the cloud which is going to replicate its data on other storage servers should ensure that they are trusted, i.e., they are members of the trusted storage system, before storing data in them. The storage node N requests the TTPN to send a list of ready nodes that accept data for replication. Since, the ready node list consists of storage nodes that are trusted, the storage node N is assured that the data will be replicated securely. After receiving the ready node list from the TTPN, the storage node N sends its data to the storage nodes, (Ni, Ni+1, Ni+2,…), in the received list.

NOTATIONS
Nid: ID of node N

{(Ni, pub(Ni)), (N(i+1), pub(N(i+1))), (N(i+2), pub(N(i+2))),……..)}: Pairs of IDs and public keys of ready nodes.
Pri(TTPN): Private key of the TTPN
EFSK: Symmetric key created and used by the encrypted file system of the storage node to encrypt/decrypt the client's data
{files}EFSK: Client's files/data encrypted with the encryption file system key
Uid: User ID
UK: User key.

Protocol 4
Message 1: {"Data Replication", Nid, nN}
Message 2: {nN, (Ni, pub(Ni)), (N(i+1), pub(N(i+1))), (N(i+2), pub(N(i+2))),……..}pri(TTPN)
Message 3: {{files}EFSK, {H({files}EFSK), EFSK, Uid, UK} pub(Ni)} //

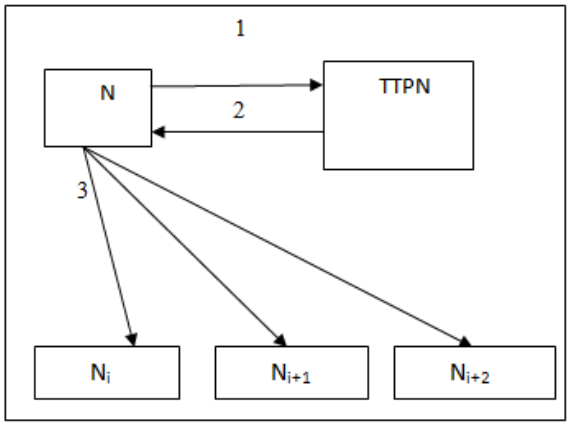


Figure 5: A trusted storage node sends its data for replication to other trusted storage nodes.

Data Access-
Protocol 5 explains how the user accesses his data from the cloud securely. The user contacts the front-end server, FES, to access his data. The FES sends the user's request to a storage node which has the user's files and the storage node securely sends the user files to the user.

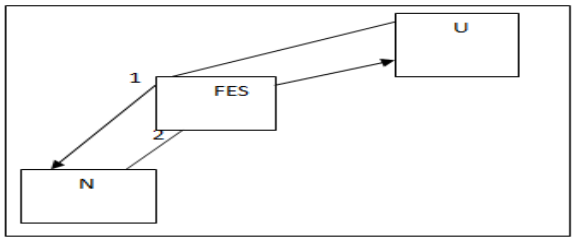


Figure 6: User accesses data.

NOTATIONS
Uid: User ID
nU: nonce of the user
files: Client's files
pub(N): Public key of the storage node N
pri(N): Private key of the storage node N
UK: User key

Protocol 5
Message 1: {"Access Data", {nU, file name/set of file names} UK, Uid}
Message 2: {{files, nU, pub(N)} UK , (nU+1) pri(N)}

The user accesses his data stored in the cloud by sending the file name or a set of file names it needs to access and the user's nonce, nU, and the user ID to the front-end server (FES) in message 1. The user encrypts the nonce and the file names with its key, UK, to get itself authenticated by the cloud so that the cloud does not send the requested files to someone else.

RESULTS ANALYSIS

In protocol 1, we show that only storage nodes with the expected platform state join the trusted storage system. A storage node prepares a measurement list, MLN, which denotes the measurement of the current state of the system at boot time. The measured state of the system is securely stored in the TPM of the storage node for maintaining the integrity of the measured state. The TTPN has the expected state of the platform stored in it. When a TTPN receives a "join system" request from the storage node, it sends a "send platform state" request to the storage node. In return, the storage node sends the measurement list, MLN, and the platform state, PCRVN, stored in the PCR registers of the TPM which is signed by the TPM. The TTPN trusts the signed content of the TPM after viewing the attestation identity key certificate which is signed by a certification authority. There is a possibility that a malicious/ compromised node can modify the measurement list, MLN, to represent a genuine platform state. The TTPN compares the signed PCR value, PCRVN, with the measurement list, MLN, of the node. If they are different, the TTPN does not attest to the storage node. The attestation fails. If they are equal, it trusts the measurement list, MLN, of the node. The TTPN now compares the measurement list, MLN, with the expected state stored within it. If they are different, the attestation fails. If they are equal, the TTPN trusts the platform running on the storage node and hence attests to the node and sends a "joined" message to the node. So, only nodes with an expected/genuine platform are allowed to join the trusted storage system.
In protocol 2, we show that a user's data is securely transmitted to the cloud and is stored on trusted storage nodes only, thus, achieving confidentiality and integrity.

In protocol 3 & 4,We show that a user's data is replicated only on trusted storage nodes. The TTPN prepares a list of storage nodes which are ready to accept data for replication. This ready node list contains trusted nodes only. A storage node sends a request to the TTPN to send the ready node list to it for replicating its data. When the TTPN receives the request, it sends the ready node list to the storage node. There is a chance that an attacker modifies the ready node list prepared by the TTPN or creates his own list of malicious nodes and sends it to the storage node N. Since message 2 is signed with the private key of the TTPN, the list cannot be tampered or changed by any attacker. Hence, the storage node is assured that the ready node list that it receives is prepared by the TTPN only. The storage node replicates its data onto the nodes in the received list. Hence, data is replicated on trusted storage nodes only and is secure.

In protocol 5 we show that the protocol allows secure data access by a user. A user U requests the cloud to access his data by encrypting the required file name/ set of file names with his user key, UK. When a user U sends a request to access his data, the storage node encrypts the user's data with the user's key, UK, and sends the encrypted data to the user. UK is known only to the user, TTPN and the trusted storage nodes. So the data is securely accessed by the user. The signature, (nU+1)pri(N), assures the user that the message is prepared by the storage node only. The user nonce in the messages ensures that the message received by the user is fresh.

CONCLUSION

The proposed framework provides data security against the system administrator who has the maximum number of privileges on a storage node. The system uses a Trusted Platform Module (TPM) chip in each storage node which provides protected storage of encryption/decryption keys of client's data and remote attestation of each storage node. Each storage server has an encrypted file system which encrypts the client's data and the corresponding keys are stored in the TPM. Cryptographic techniques are used to provide secure communication between the client and the cloud. The system ensures that the client's data is stored only on trusted storage servers and it cannot be transferred by malicious system administrators to some corrupt node.

The system architecture and the proposed protocols together form a Trusted Storage System for Cloud. The system achieves confidentiality and integrity of the client's data stored in the cloud. Storage server has an encrypted file system which encrypts the client's data and the corresponding keys are stored in the TPM. Cryptographic techniques are used to provide secure communication between the client and the cloud. The system ensures that the client's data is stored only on trusted storage servers and it cannot be transferred by malicious system administrators to some corrupt node. The system architecture and the proposed protocols together form a Trusted Storage System for Cloud. The system achieves confidentiality and integrity of the client's data stored in the cloud.

REFERENCES

[1]. Muhammad Aufeef Chauhan and Muhammad Ali Babar, "Migrating Service-Oriented System to Cloud Computing: An Experience Report", 2011 IEEE 4th International Conference on Cloud Computing, pp 404-411.
[2]. Louridas, P., Up in the Air: Moving Your Applications to the Cloud. Software, IEEE, 2010. 27(4): p. 6-11.
[3]. Ali Babar, M., Chauhan, M. A., A Tale of Migration to Cloud Computing for Sharing Experiences and Observations, SECLOUD workshop, Collocated with ICSE 2011, Hawaii, USA.
[4]. RajkumarBuyyaa, Chee Shin Yeoa, , SrikumarVenugopala, James Brobergа, and IvonaBrandicc, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future Generation Computer Systems, Volume 25, Issue 6, June 2009, Pp 599-616.

[5]. K. Keahey and T. Freeman, "Science Clouds: Early Experiences in Cloud Computing for Scientific Applications," in proceedings of Cloud Computing and Its Applications 2008, Chicago, IL. 2008.
[6]. JunjiePeng, Xuejun Zhang, and Zhou Lei, Bofeng Zhang, Wu Zhang, Qing Li, "Comparison of Several Cloud Computing Platforms", Second International Symposium on Information Science and Engineering, IEEE 2009, pp 23-27.
[7]. Ommeren, E. V., Duivestein, S., deVadoss, J,Reijnen, C. &Gunvaldson, E. Collaboration in the Cloud. Microsoft and Sogeti, Bariet, Ruinen, the Netherlands, 2009.
[8]. Oram, A. "Cloud computing perspectives and questions at the World Economic Forum," WikiContent, 2009. Accessed January 20, 2011