Securing SaaS Cloud Infrastructure using TPM based provisioning

Mr. Pramod[1], Dr. B. R. Prasad Babu[2]

Abstract: In the era of technology, cloud computing is the vast developing technology. The research indicates that as cloud elements are becoming more extensive in IT industry, more enterprises are depending on cloud computing for their business needs such as storage, business disaster recovery and security. Thus, cloud computing is almost accepted as a default part of IT landscape with the indicators of this strong usage and strong market. Software as a service (SaaS) is software model the users can access the applications that is owned, delivered and managed remotely by providers. The key advantage to SaaS in the enterprise is in cost savings - in personnel, in hardware and in physical storage space. However, based on data sharing properties, these may be vulnerable to malicious attacks. Thus, with the user credentials it can be easily compromised and the services of SaaS are accessed. The services can be acquired even by the URLs when compromised.

In order to overcome this issue we propose a novel technique, Securing SaaS Cloud Infrastructure using TPM based provisioning. A portable TPM is used for accessing SaaS which provides better security. A cryptographic protocol that enables the remote authentication which preserves the privacy of the user is modelled as Trusted Platform Module (TPM). TPM is used for strong user authentication framework apart from user credentials which proves the secure data access control in the cloud storage by providing additional security. Also, our system is constructed based on the cloud MVC architecture. MVC is significant, which allow fast & agile development and provide full control over mark-up. Hence it is finest for establishing interactive web application. The scripts in this model are run only when it is required, thus reducing the overall computation. Using this approach, services are provided to the user in an efficient and secured way. Finally, we demonstrate the secured services and efficiency of the proposed schemes through extensive experimental evaluation on the live Microsoft Windows Azure platform.

Keywords: TPM, Azure, Cloud MVC Architecture, SaaS, Symmetric Encryption.

I. INTRODUCTION

LOUD computing is a technology that uses the internet and central remote servers to maintain data and applications. Cloud computing allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access. Gartner describes cloud computing as a style of computing in which scalable and elastic IT-enabled capabilities are delivered “as a service” using Internet technologies [5]. It has also become a hot industry term that has been used in many contradictory ways. The use of cloud computing is growing, and by 2016 this growth will increase to become the bulk of new IT spend, according to Gartner, Inc. 2016 will be a defining year for cloud as private cloud begins to give way to hybrid cloud, and nearly half of large enterprises will have hybrid cloud deployments by the end of 2017.“In India, cloud services revenue is projected to have a five-year projected compound annual growth rate of 33.2 percent from 2012 through 2017 across all segments of the cloud computing market. Segments such as software as a service (SaaS) and infrastructure as a service (IaaS) have even higher projected CAGR growth rates of 34.4 percent and 39.8 percent,” said Ed Anderson, research director at Gartner.“Services delivered through the cloud will foster an economy based on delivery and consumption of everything from storage to computation to video to finance deduction management,” said Chris Howard, research vice president at Gartner.

SaaS (software as a service) [1] provisioning refers to the process for on-boarding or establishing service delivery to users of cloud based software applications.Integration remains a major concern for those hesitant to adopt SaaS, but a recent vendor survey shows a new top barrier: data privacy [12]. Thus, accessing the data with preserving the

About 1st Mr. PRAMOD, Research Scholar, VTU, EPCET, Bangalore -49
(9845802330, email: pramod741231@gmail.com)
About 2nd Dr. B R PRASAD BABU, Head of Department, Computer Science and Engineering, SEAIT, Bangalore-49.
(08951516144, email: brprasadbabu@gmail.com

confidentiality is the main concern. The services provided by the SaaS can also be accessed by the attackers if the user details are leaked. Thus, security must be provided in such a way that, the credentials cannot be compromised by the malicious attackers so as to access the services provided by the SaaS.Although there are numerous benefits of a SaaS model there are also some drawbacks to consider. As data is stored on cloud, security becomes a major issue and also the latency issue.

The existing integrity attestation scheme providing service for SaaS clouds mechanisms [14] are not able to process a portable secure based service framework in the multitenant cloud system. Even though it achieves more accurate pin pointing than other existing schemes under strategically colluding attacks, it is still not adaptable when concerned with the user confidentiality being compromised. The need for an efficient high securable in accessing the SaaS Cloud infrastructure is needed. Thus, this paper aims at providing such a role mechanism. In this paper, we have proposed Securing SaaS Cloud Infrastructure using TPM based provisioning. This is an extension of Portable TPM Based User Attestation Architecture for Cloud Environments where security is highlighted in the cloud infrastructure. This TPM [15] is a notion of trust from the perspective of a service provider's control over data and a registered user. Due to remote attestation protocol for verifying the client, we ensure that malicious behaviours cannot occur. The TPM is a link between a cloud service provider and a registered user through the authenticator. Therefore, a user can access to cloud storage's contents in secure environment and store user data to the remote server in encrypted form using securely created and managed data encryption key(TPM). Services are provided for the cloud users with respect to the TPM. Thus, TPM enhances in securing the user from being attacked by the outsiders. Additionally, it is important that the services should not be accessed by the attackers with the URLs. The services provided by the cloud providers for the users are dynamic in nature. Here, we have implemented the cloud MVC architecture. This provides a consolation of the application's presentation layer that exhibit the information in the user interface, from the way the information is actually processed so as to provide better code organization, expandability, scalability and code re-use.

The main contributions of this paper are thus summarized as follows. Creating a secured cloud environment for the users with the help of portable TPM. Formulating the execution overhead for service parameter such as creating containers and providing the cloud CPU utilization by taking into account both the cloud providers and the cloud users, while preserving the confidentiality of the data when accessing the SaaS. Providing a $\mathbf{Q_OS}$ perspective for SaaS Cloud memory utilization in creating number of containers with the presence of TPM to enable the efficient usage processing and also proving through extensive tests that this approach is applicable to public clouds.

The research paper is organized as follows. Section two discusses the related work. Our proposed model is presented in section three. The experimental results and comparisons are presented in section four. The concluding remarks are discussed in the last section of the paper.

II. RELATED WORK

Much work has been done in service provisioning in SaaS based on portable TPM in Cloud Computing sector. Let us look into some of the survey which exists. In [14], it presents a scalable and effective service integrity attestation framework for SaaS clouds which provides stronger attacker pinpointing power techniques. A prototype is implemented and tested on cloud computing infrastructure using IBM System S stream processing applications. This scheme experimental results show that it can achieve higher attacker pinpointing accuracy than existing approaches. This approach does not require any special hardware or secure kernel support and imposes little performance impact to the application, which makes it practical for large-scale cloud systems. Even though it is scalable, it has some limitations. First, malicious attackers can still escape the detection if they only attack a few service functions, take majority in all the compromised service functions, and have less inconsistency links than benign service providers. However, it can effectively limit the attack scope and make it difficult to attack popular service functions. Second, this approach needs to assume the attested services are input deterministic where benign services will return the same or similar results defined by a distance function for the same input. Thus, this scheme cannot support those service functions whose results vary significantly based on some random numbers or time stamps.

A middleware-layer that handles the authentication process on behalf of the consumer devices in real time and with minimal HTTP traffic is been introduced in [2]. Here this approach is designed for the mobile users to access IaaS cloud services from Amazon S3, Drop box, and MEGA in soft real time. The primary goal of this work is to relieve the mobile device from the tedious authentication process and further shield the existence of the IaaS cloud sources from the mobile consumer. The flexibility of this scheme also includes authentication through social networking services such as Facebook, Google+, Twitter, and Yahoo ID based on the O Auth 2.0 technique. Thus, focusing only on the authentication process. This framework will not support the IaaS services composition. Currently, the IaaS services are considered as separate individual services and this method only handles the authentication process.

The paper [3] presents a SaaS application service provisioning problem with respect to users. For SaaS provider, it hopes that it can provide better service performance to tenants while attain more profit. But the two goals are contradictory. Thus, it effectively achieve a trade-off of profit and service performance. But, it is not considering the service performance factors of storage space, network bandwidth, data security, and cost factors of data transmission cost. Lianfen Huang and et al. [21] refers Long-Term Evolution (LTE) network which is next generation network beyond 3G.They use EAP-TLS which provide robust security if the network user are not very concerned with the overhead. Tsu-Yang Wu and et al. [20] proposed revocable 10- based signature scheme with batch verification. In these, the exiting user authentication schemes have many security flaws. An In-Out-VM dynamic measurement architecture for virtual machine (VM),which aims at user’s running applications rather than static executable files is presented in [4]. It detects dynamic attacks and supports fine-grained protection such as measuring the code segment and the argument segment separately. It is implemented by a hybrid of In-VM method and Out-of-VM method. The implementation is given equipped with the Trusted Platform Module (TPM). Only one of the available PCRs is occupied in this design to save limited resources. The Platform Configure Registers are limited in the TPM which is embedded.

[6] Uses Identity-based techniques for mutual authentication in the network or infrastructure called private clouds. This generates a shared key or group key for mutual Authentication and secure communication. This scheme divides the sharing users into the very same domain and in this domain relies on the sharing group secret key to exercise mutual authentication. By the analysis of performance, this scheme improves the computational and communicational efficiency. In ID based mutual authentication scheme, the group key of same domains, it reduces the no of key requires for total communication. If group key is breach then security will be exploiting, so it must be transfer in a secure manner. The paper [9] specifies data protection requirements and proposed Biometric Authentication as a Service for strong authentication in web environments based on the Software as a Service model. Thus, for providing much privacy and reducing data protection risks. Prototypical implementation of a SaaS-compliant biometric authentication service based on keystroke dynamics for enterprise deployment is given. In public cloud applications, due to open accessibility, additional security- and performance related risks must be taken into account. Thus, enhancement of interfaces and security controls are to be considered. In the paper [10], the problem of data storage security in cloud computing which is essentially a distributed storage system are examined and a third party auditor scheme is proposed. The advantage of this scheme is the cloud service provider can offer the functions which were provided by the traditional third party auditor and make it trustful. To ensure each data access in control and reduce the complexity of cloud computing, it proposes a scheme using RSA and Bilinear Diffie-Hellman techniques to accomplish the authentication function with the minimum cost. Confidentiality of users’ access privilege and authentication accountability can be achieved. But still, if the third party revels or compromised then this scheme is found to be unsecure.

A protocol for the deployment of a data owner-generated Trust Ticket is devised in [11].Trust Ticket is a link between a cloud service provider and a registered user through a data owner. A data owner encrypts the data with secret key and outsources the encrypted data to a cloud service provider. A data owner also updates that data. A data owner remains online only during a user’s registration. Unless a data owner makes any changes, a registered user uses the Trust Ticket and the secret key for a data owner’s data service from a cloud service provider. However, devising a mechanism to counter a malicious situation of a user’s with a cloud service provider should be made. In [13], a distributed secure collaboration framework for cloud collaboration service is given. The cloud vendor maps the requested permissions into appropriate local roles in order to allow resource access. However, coexistence of multiple simultaneous access requests may introduce conflicts which violate the principle of security. This module uses only local information to detect conflicts and remove them. Therefore, this work has to be extended to multi-cloud federated environment, where the issue of semantic heterogeneity will be addressed along with issues of security and availability.

In paper[7], proposes a framework that preserves privacy in the context of highly customizable composite web services. This approach involves service producers exchanging their terms-of-use with consumers in the form of models. The compatibilities between privacy policies and service models can be verified automatically at the user end using the techniques proposed in this paper. Any conflicts can result in obligations that are provided to the producer, who can enforce these obligations using dynamic analysis techniques that have proposed here. Thus, not providing much significance on services. [8] Proposes a privacy proxy service (PPS)-based architecture to enhance user data privacy in service compositions with nested Web services. [16] This design does not prevent services from colluding. But this proposal should be designed in order to work in cloud computing. [18], [19] techniques need to be scaled upon a special hardware which is trusted or at least needs a kernel support which is secure. But this approach is difficult when trying to deploy on large-scale cloud computing infrastructures. To overcome all these issues, we have proposed a system Securing SaaS Cloud Infrastructure using TPM based provisioning which is secure and can access the services in the public cloud. In this model portable TPM is designed and a high performance access control service provisioning mechanism is proposed.

III. PROPOSED SYSTEM

Before we understand the service provisioning in SaaS cloud architecture, let us know the working of TPM and how the user is getting registered with the TPM which provides additional authenticated security. The working of portable TPM is the same as that of the previous work - Portable TPM based user attestation architecture for cloud environments. Thus, the authentication and verifying phases are discussed here with respect to the registration and login of the user. The blacklisting cases are mentioned. The whole system is build using the Cloud MVC architecture. Thus, a brief detail is given on the same. Later, the service provisioning is discussed for the cloud user when authenticated with the TPM.

PORTABLE TPM

Here, a cloud provider, cloud users, authenticator and the cloud verifiers are concerned. The membership certificates for the cloud users are issued by the cloud provider. Membership certificates are blacklisted by the blacklisting controller (from verifier). The cloud users in the system may vary and also users may access their data according to their need. Let us consider a hardware based authentication key in an ideal system. The operation carried out by the authentication key $\mathbb{K}$ are initialize, register, membership approval and blacklisting.

In initialize phase, every entity is controlled by the controller which is indicated by the authentication key. Users are need to be registered. A user requests the authenticator with $\mathbb{K}$ andthe authenticator asks the cloud provider whether the usercan get registered. If the cloud provider agrees, the authenticator notifies the user that he can become a member. In the membership approval phase, the authenticator sends a request that he wants to contact the verifier. With $\mathbb{K}$, it informs the verifier that user wants to perform the membership approval without revealing to the verifier who the authenticator is. The verifier chooses a messages andsends s to the authenticator. If the authenticator is not a member,$\mathbb{K}$ aborts. Otherwise, $\mathbb{K}$ tellsthe authenticator whether he has been blacklisted and askshim whether to proceed. If the authenticator does not abort, $\mathbb{K}$ lets the verifier know that a blacklisted user has signed the message s.Otherwise, $\mathbb{K}$ informs the verifier that s has been signed by a legitimate member.Blacklist revokes the membership authentication. The blacklisting controller tells the authenticator to blacklist a user. If the user is not a group member,$\mathbb{K}$ denies the request. Otherwise, $\mathbb{K}$ marks the user as blacklisted.

A user who is not a member or is a member but has been blacklisted cannot succeed in membership approval toany verifiers. The verifier cannot identify who is the authenticator in a membership approval operation, thus proving anonymity. Blacklist causes verifiers to reject messages signed by a blacklisted user in an ideal system. In our protocol, if a user’s private key is exposed and the cloud user is blacklisted, the signatures from this blacklisted cloud user become linkable to an honest verifier. As a result, corrupted users who reveal their private keys and are blacklisted deliberately lose their privacy. Thus, an authenticator can check whether the user has been blacklisted from on the blacklist, before the user signs a signature and sends it to the verifier. If the authenticator finds out that the user has been blacklisted, he can choose to not proceed. The security of our scheme relies on the public key cryptographic protocol and the Diffie-Hellman assumption.

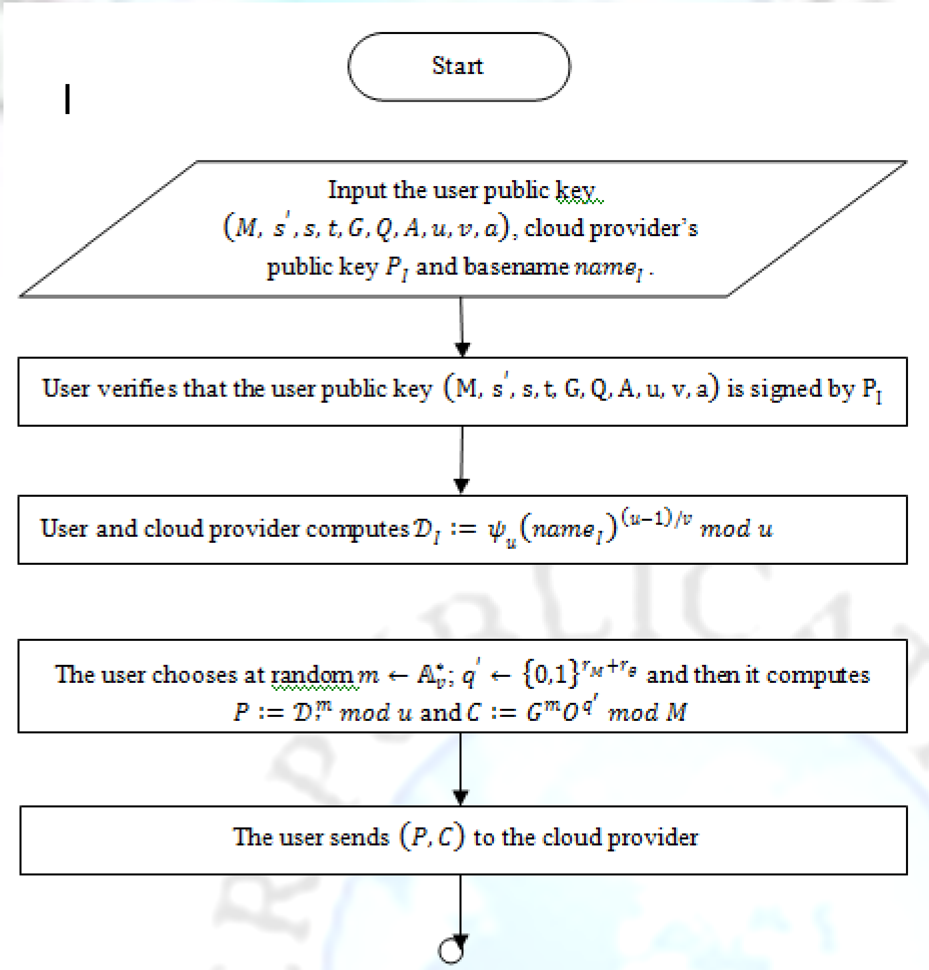
Remote authentication of the hardware based authentication key is enabled in the cryptographic protocols. Here, it preserves the privacy of the cloud user which contains the key$\mathbb{K}$. This protocol consists of the cloud provider, authenticator who provides access issued by the cloud provider and the verifier who verifies with the authenticator. The authenticator consists of the portable key $\mathbb{K}$ which preserves the privacy for the cloud user.

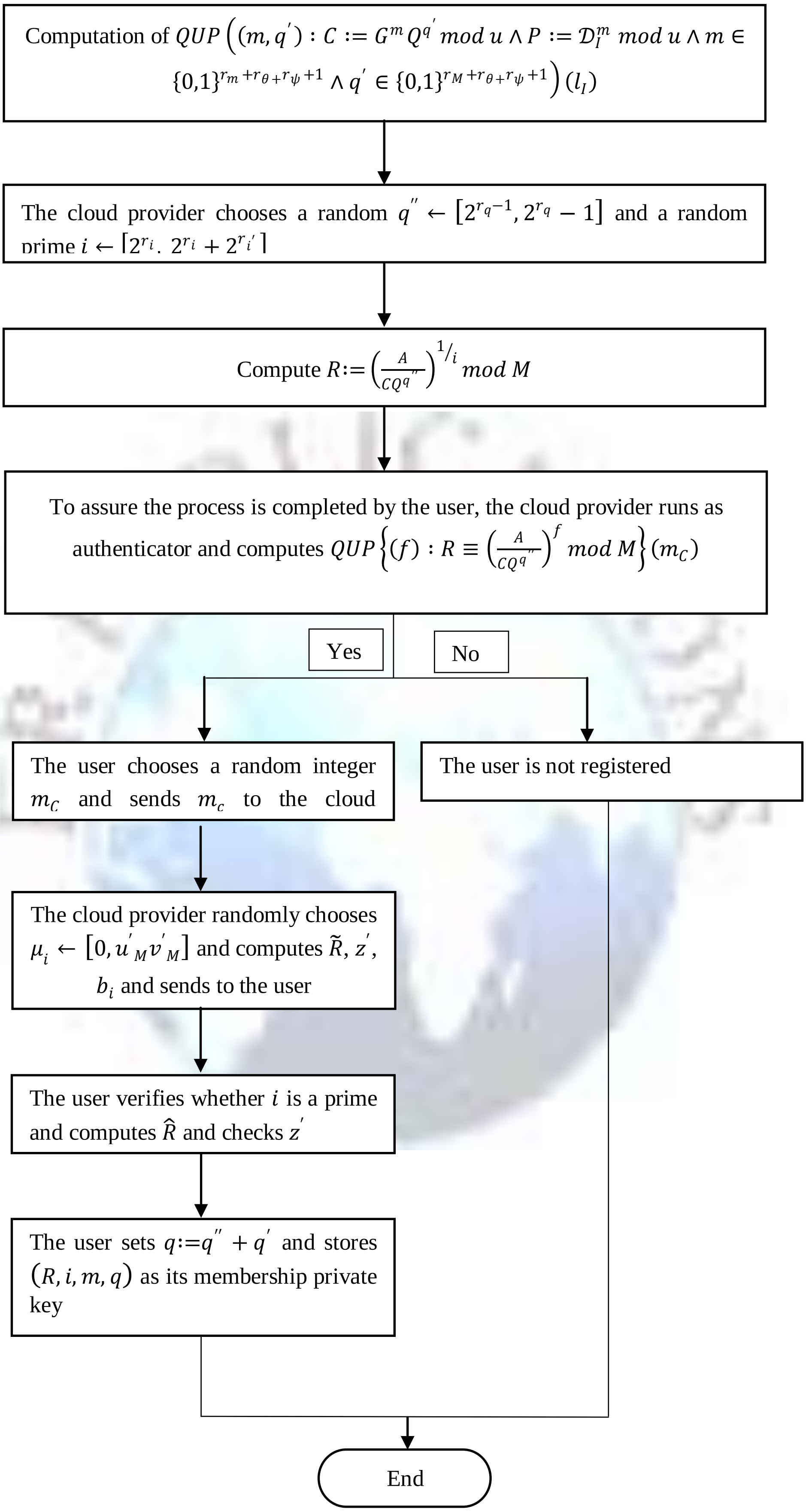
The key generation program also produces a non-interactive proof that the public key was formed correctly. Here we describe how the cloud provider chooses the public key and the user issuing private key. The later will guarantee the security properties, i.e., that privacy and anonymity of signatures will hold. The cloud provider chooses a public-key cryptographic modulus$\mathrm{M} = \mathrm{u_M v_M}$ with $\mathrm{u_M} = 2\mathrm{u'_M} + 1, \mathrm{v_M} = 2\mathrm{v'_M} + 1$ such that $\mathrm{u_M}, \mathrm{u'_M}, \mathrm{v_M}, \mathrm{v'_M}$are all primes,$\mathrm{u_M}$ and $\mathrm{v_M}$ have the same length, andm has $\mathrm{r_M}$bits.

Furthermore, the cloud provider chooses a random generator$\mathrm{s'}$of $\mathrm{ZG_M}$, the group of quadratic residues modulo M. Next, it chooses random integer$\mathrm{e_s}, \mathrm{e_t}, \mathrm{e_q}, \mathrm{e_b}, \mathrm{e_g} \in [1, \mathrm{u'_M v'_M}]$ and computes$\mathrm{s, t, G, Q}$and A.Finally, the cloud provider publishes the public key$(\mathrm{M, s', s, t, G, Q, A, u, v, a})$and the proof, and stores$(\mathrm{u'_M}, \mathrm{v'_M})$ as the user issuing private key. In addition to generating the user public key and user issuing private key, the cloud provider generates also a longtermpublicprivate key pair$(\mathrm{P_I}, \mathrm{P_I^{-1}})$. The cloud provider publishes the public key P. This key is used for authentication between the cloud provider and any user who wants to become a registered member. The flow diagram for the registration phase, user membership approval and the blacklisting conditions are given below.

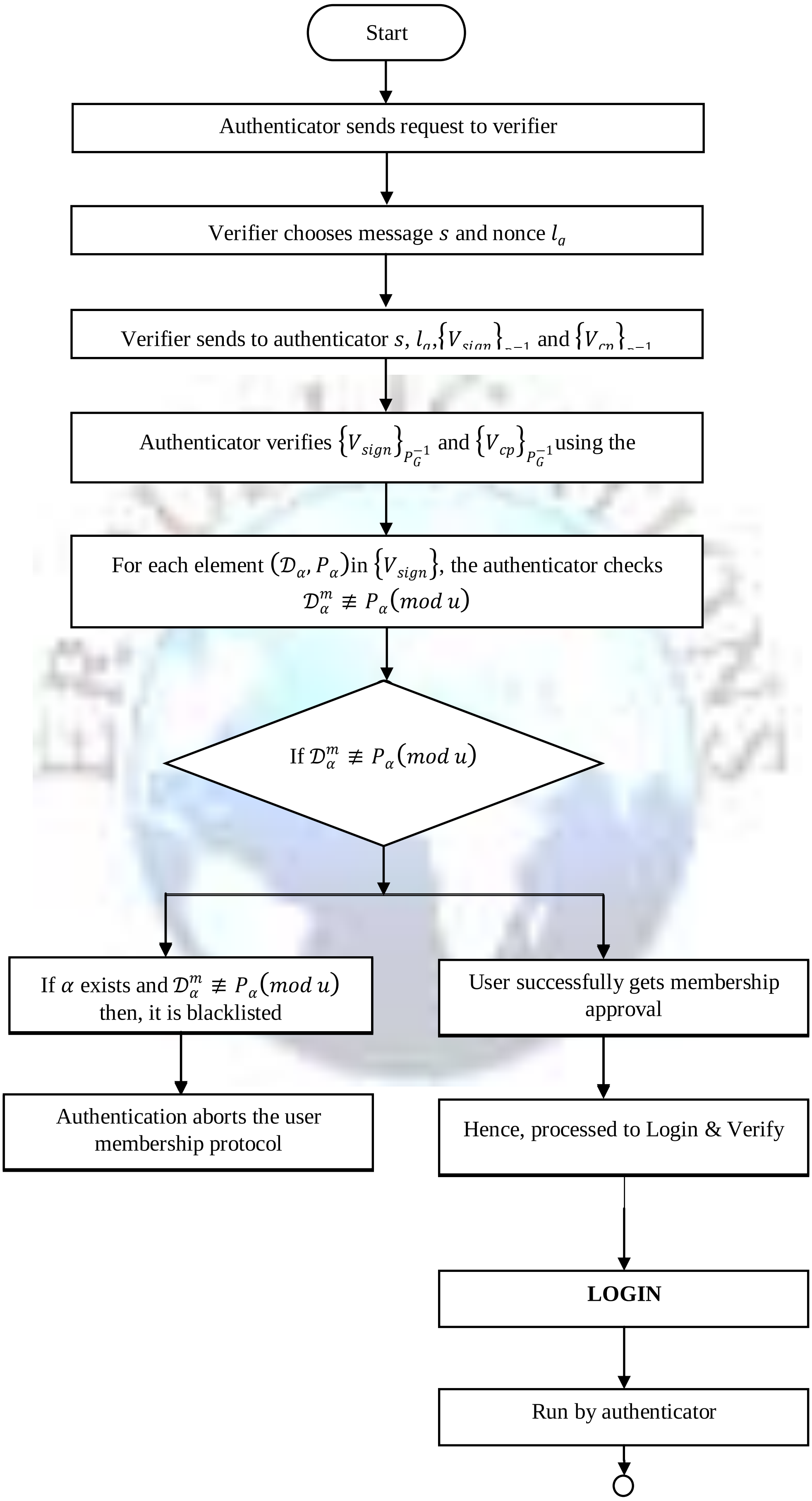
I. Registration Phase

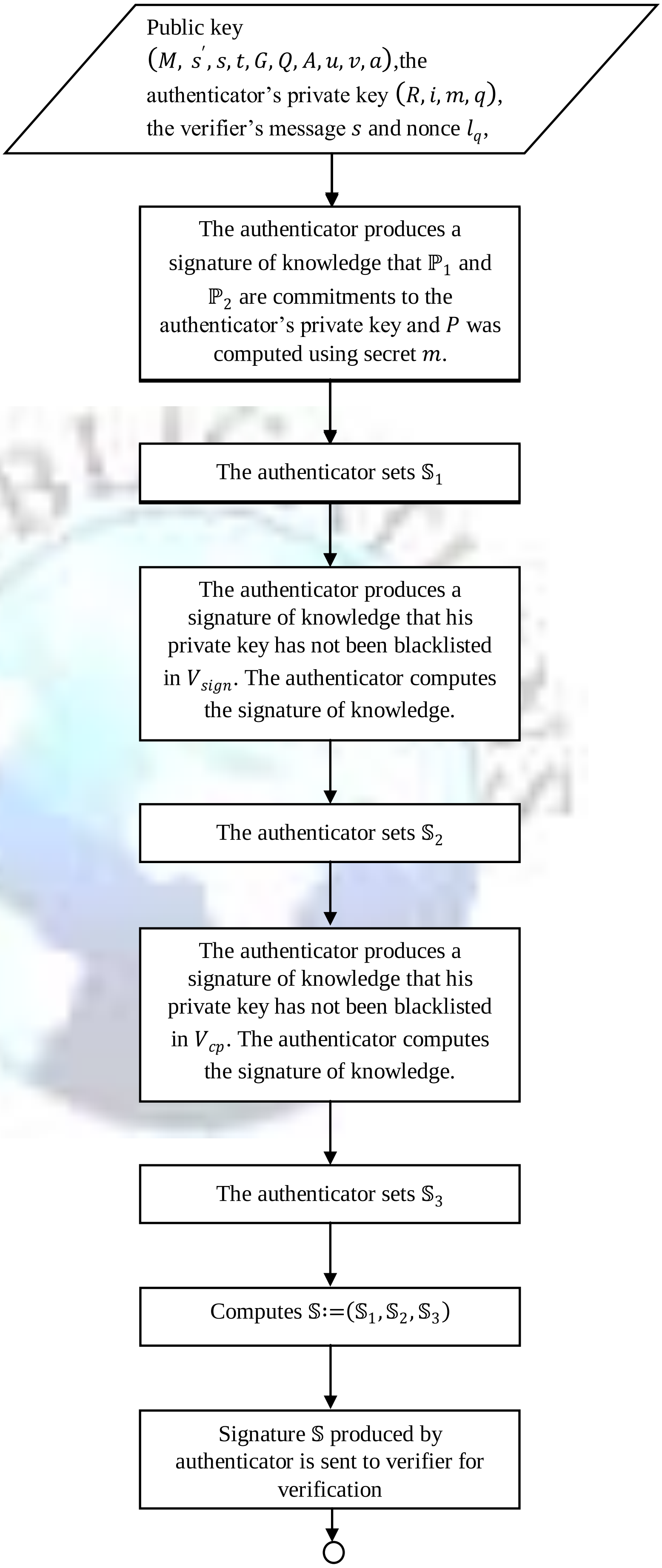
The complete registration process for the user to get registered with the cloud services is represented in the flowchart. This involves cloud user, cloud provider, authenticator and the verifier. The user registers with the help of TPM.

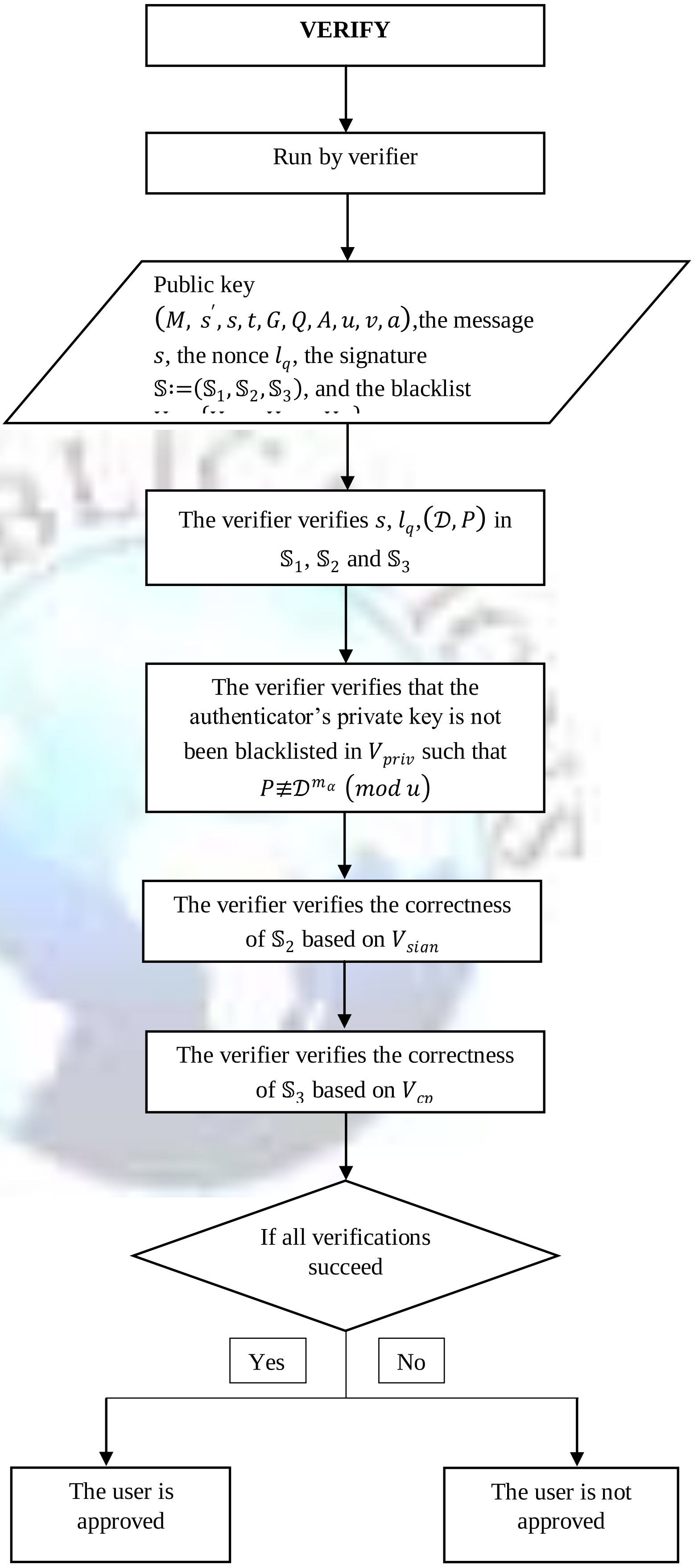

Start
Input the user public key $(M, s', s, t, G, Q, A, u, v, a)$, cloud provider's public key P_I and basename $name_I$.
User verifies that the user public key $(\mathrm{M, s', s, t, G, Q, A, u, v, a})$ is signed by $\mathrm{P_I}$
User and cloud provider computes $\mathcal{D}_I := \psi_u(name_I)^{(u-1)/v}$ $mod\ u$
The user chooses at random $m \leftarrow \mathbb{A}_v^*$; $q' \leftarrow \{0,1\}^{r_M + r_e}$ and then it computes $P := \mathcal{D}_I^m$ $mod\ u$ and $C := G^m O^{q'}$ $mod\ M$
The user sends (P, C) to the cloud provider


Computation of $QUP\left((m,q'):C:=G^{m}Q^{q'}\ mod\ u\wedge P:=\mathcal{D}_I^{m}\ mod\ u\wedge m\in\{0,1\}^{r_m+r_\theta+r_\psi+1}\wedge q'\in\{0,1\}^{r_M+r_\theta+r_\psi+1}\right)(l_I)$
The cloud provider chooses a random $q''\leftarrow[2^{r_q-1},2^{r_q}-1]$ and a random prime $i\leftarrow[2^{r_i},\ 2^{r_i}+2^{r_i'}]$
Compute $R:=\left(\frac{A}{CQ^{q''}}\right)^{1/i}\ mod\ M$
To assure the process is completed by the user, the cloud provider runs as authenticator and computes $QUP\left\{(f):R\equiv\left(\frac{A}{CQ^{q''}}\right)^{f}\ mod\ M\right\}(m_C)$
Yes
No
The user chooses a random integer m_C and sends m_c to the cloud
The user is not registered
The cloud provider randomly chooses $\mu_i\leftarrow[0,u'_Mv'_M]$ and computes $\tilde{R}$, z', b_i and sends to the user
The user verifies whether i is a prime and computes $\hat{R}$ and checks z'
The user sets $q:=q''+q'$ and stores (R,i,m,q) as its membership private key
End

User Membership Approval Protocol

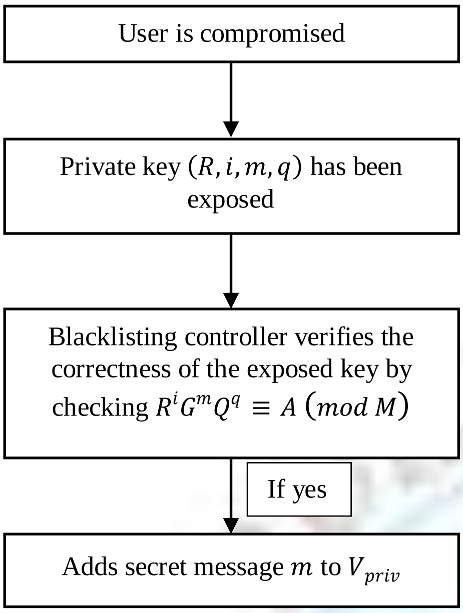

Start
Authenticator sends request to verifier
Verifier chooses message s and nonce l_a
Verifier sends to authenticator s, l_a,$\{V_{sign}\}_{P^{-1}}$ and $\{V_{cp}\}_{P^{-1}}$
Authenticator verifies $\{V_{sign}\}_{P_G^{-1}}$ and $\{V_{cp}\}_{P_G^{-1}}$using the
For each element $(\mathcal{D}_\alpha, P_\alpha)$in $\{V_{sign}\}$, the authenticator checks $\mathcal{D}_\alpha^m \not\equiv P_\alpha (mod\, u)$
If $\mathcal{D}_\alpha^m \not\equiv P_\alpha (mod\, u)$
If α exists and $\mathcal{D}_\alpha^m \not\equiv P_\alpha (mod\, u)$ then, it is blacklisted
User successfully gets membership approval
Authentication aborts the user membership protocol
Hence, processed to Login & Verify
LOGIN
Run by authenticator


Public key $(M, s', s, t, G, Q, A, u, v, a)$,the authenticator's private key (R, i, m, q), the verifier's message s and nonce l_q,
The authenticator produces a signature of knowledge that $\mathbb{P}_1$ and $\mathbb{P}_2$ are commitments to the authenticator's private key and P was computed using secret m.
The authenticator sets $\mathbb{S}_1$
The authenticator produces a signature of knowledge that his private key has not been blacklisted in V_{sign}. The authenticator computes the signature of knowledge.
The authenticator sets $\mathbb{S}_2$
The authenticator produces a signature of knowledge that his private key has not been blacklisted in V_{cp}. The authenticator computes the signature of knowledge.
The authenticator sets $\mathbb{S}_3$
Computes $\mathbb{S}:=(\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3)$
Signature $\mathbb{S}$ produced by authenticator is sent to verifier for verification

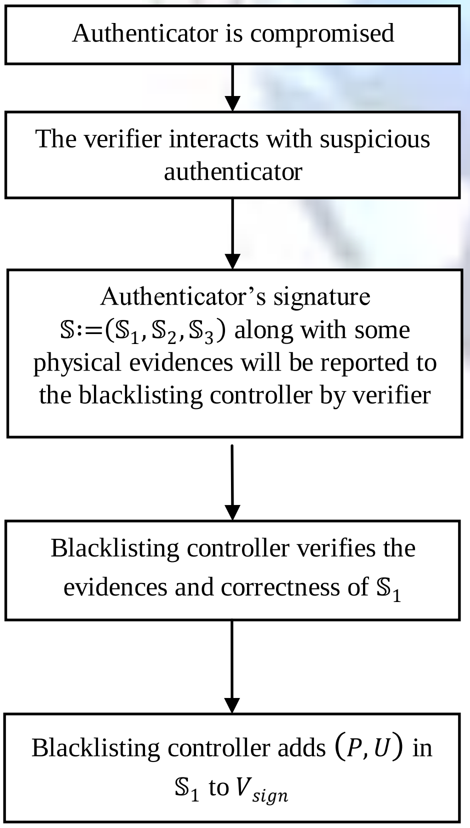

VERIFY
Run by verifier
Public key $(M, s^{'}, s, t, G, Q, A, u, v, a)$,the message s, the nonce l_q, the signature $\mathbb{S}:=(\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3)$, and the blacklist
The verifier verifies $s, l_q, (\mathcal{D}, P)$ in $\mathbb{S}_1$, $\mathbb{S}_2$ and $\mathbb{S}_3$
The verifier verifies that the authenticator's private key is not been blacklisted in V_{priv} such that $P \not\equiv \mathcal{D}^{m_\alpha}\ (mod\ u)$
The verifier verifies the correctness of $\mathbb{S}_2$ based on V_{sian}
The verifier verifies the correctness of $\mathbb{S}_3$ based on V_{cn}
If all verifications succeed
Yes
No
The user is approved
The user is not approved

BLACKLIST

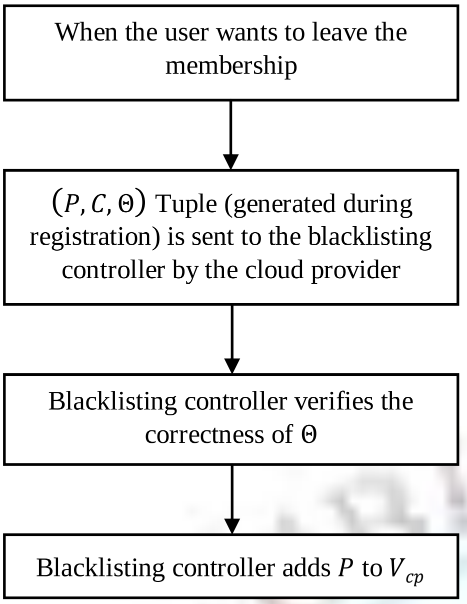
Case 1. When the user is compromised.


User is compromised
Private key (R, i, m, q) has been exposed
Blacklisting controller verifies the correctness of the exposed key by checking $R^i G^m Q^q \equiv A \ (mod\ M)$
If yes
Adds secret message m to V_{priv}

Case 2. When the authenticator is compromised.


Authenticator is compromised
The verifier interacts with suspicious authenticator
Authenticator's signature $\mathbb{S}:=(\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3)$ along with some physical evidences will be reported to the blacklisting controller by verifier
Blacklisting controller verifies the evidences and correctness of $\mathbb{S}_1$
Blacklisting controller adds (P, U) in $\mathbb{S}_1$ to V_{sign}

Case 3. When cloud provider wants to blacklist a user.


When the user wants to leave the membership
(P, C, Θ) Tuple (generated during registration) is sent to the blacklisting controller by the cloud provider
Blacklisting controller verifies the correctness of Θ
Blacklisting controller adds P to V_{cp}

SERVICE PROVISIONING

MVC Architecture

In the implementation, the Model–View–Controller (MVC) approach is adopted in order to enforce separation of concern. The MVC approach also aided us to enforce distributed transparency when necessary as well as the need to hide the abstraction of the IaaS clouds from the cloud users. The View component is built using HTML5 and this is the GUI part that is deployed. The communication between the cloud user and the controller is through the Http Web Request Class in C#. The Controllers are the main application server components that are exposed to the Views and the Models. The controller performs roles related to the user register, login, verify, authentication, storage and request routing to/from the cloud sources. The model and controller are built in C# using the .Net framework. Currently, Azure has Microsoft Azure Development Kits that fully support .Net and other standardized programming environments. The whole framework is deployed following the REST standard for data manipulation and requesting.

Software-as-a-service is a software delivery design in which software and its operation are accessed remotely based on the web. In this web-based model, software vendors host and maintain the servers, databases, code that constitute an application and deliver their applications via large-scale cloud computing infrastructures. Thus, allowing the cloud users to access cloud hosted software and applications. Microsoft Windows Azure, Amazon Web Service and Google App Engine are few providers who deliver the application services so as to support business applications and big data processing. Securing the data is one of the major concern considering SaaS. When granting someone else to maintain data especially in huge enterprise with large data set, security plays a major role.

Here, we focus on the peculiar way where existing user cloud providers from the IaaS layer enforce security in their environments: Microsoft Windows Azure, Amazon Simple Storage Service (Amazon S3), Dropbox, and MEGA. These IaaS layers require the data user (requester) to provide credentials such as access key, secret access key and sometimes session id in order to access the stored digital assets which are mostly files and documents. While these security policies may not be an issue in a wired environment, it poses great concern for communication latency in a distributed network.

The three IaaS cloud services are all utilized as file and documents storages. The employment of these IaaS layers is on the rise recently largely because of the dawn of big data on enterprises. Moreover, most of the data being generated is unstructured which makes such NoSQL and file storages affordable as containers. Further, the services facilitate the storage of large amount of data based on “pay according to usage” policy and also support data in multiple formats and standards. There is a percentage of usage space that is free for each of the IaaS layer but customers can always pay for more space. Firstly, consider Microsoft Windows Azure for the clarity of the user and show how the security workflow is adopted by the other two IaaS providers. A data container which is called “bucket” has to be created first within which the file contents are deposited on Microsoft Azure. The service also allows the provider to specify customized metadata of every file uploaded; a feature that encourages the integration of Microsoft Windows Azure. Furthermore, due to its flexibility, Microsoft Windows Azure can be used in a composite enterprise architecture that has other cloud framework components such as Google App Engine (GAE) and Amazon S3. However, we focus on the peculiar way Microsoft Windows Azure enforces security access for data consumption.

The Microsoft Windows Azure facility follows the strict security policies laid down within the Azure data protection policy. The service permits hierarchical access to files including usage permissions (read, create, and modify operations)which well fit into enterprise oriented workflows. Based on the Microsoft Windows Azure user roles can be defined.

While the security policies promise consumer satisfaction in terms of data safety and protection, it poses other challenges too. Every user(requester) needs to have a unique Id and a secret password which are assigned by the Microsoft Windows Azure service when the users are created. Then, a Hash Message Authentication Code signature has to be generated with these credentials; which has to be added to the HTTP request headers for Microsoft Azure to authenticate the requester. Based on the signature, Azure is also able to determine the access level privileges of the requester.

In this paper, we present TPM based SaaS service provisioning, a new service providing attestation framework for the cloud architecture. TPM based SaaS service provisioning provides a practical service integrity attestation method which do not consider the third party service providing entities. Though, in the large scale cloud architectures many malicious attackers may drive colluding attacks on the service functionalities. To address the challenge, TPM based SaaS service provisioning takes a holistic approach by providing the services dynamically based on the authentication of TPM. With the username and password, the services can be accessed by malicious attackers by compromising the user credentials. In order to overcome this, portable TPM have been provided to users for further security where every user is authenticated along with his TPM. The authentication will happen only when the potable TPM device is found. Once registered user is about to access the services, the user credentials along with the TPM key is been encrypted altogether. And then it is been hashed. So when the user is trying to access the services from the cloud provider, the hashed data of the user is been mapped with the authenticator's hashed data. Only if the hashed data is verified from the verifier, the services can be accessed by the user. Else, it is declined. Therefore, meeting the security issues. If any malicious attackers is encountered, it is been backlisted as discussed above. By taking this unified approach, TPM based SaaS service provisioning can not only diagnose attackers more efficiently but also provide the cloud services in an efficient and secured way for large-scale cloud computing infrastructures.

The process is as follows. Initially for the cloud user to access the services, the user must be a registered user. The registration process takes place with the credentials along with the unique portable TPM provided to user. Refer the flowchart of registration phase using TPM. Once he user is registered, the user can login based on the credentials along with the TPM device. If the login is successful (discussed above), the user can access the services of the cloud. The services are provided for the user only if the TPM is found to be present which is evaluated as follows.

This is run by an authenticator and a verifier. The input to this program is the public key,$(M, s', s, t, G, Q, A, u, v, a)$, the authenticator'sprivate key(R, i, m, q), the verifier's message s and noncel_q.Firstly, the authenticator picks a random $\mathcal{D} \leftarrow \langle a \rangle$ and two integers $\mathbb{O}, \mu \leftarrow \{0,1\}^{r_M + r_\theta}$ and computes $\mathbb{P}_1 := Rt^{\mathbb{O}} \bmod M$, $\mathbb{P}_2 := s^{\mathbb{O}} t^{i} (s')^{\mu} \bmod M$, $P := \mathcal{D}^{m} \bmod u$
Then, the authenticator produces a signature of knowledge that$\mathbb{P}_1$ and $\mathbb{P}_2$are commitments to the authenticator's private key and P was computed using the authenticator's secret m. That is, the authenticator computes the signature of knowledge. The authenticator sets and computes $\mathbb{S} := (\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3)$. Signature $\mathbb{S}$ produced by authenticator is sent to verifier for verification. The verifier verifies s, l_q,$(\mathcal{D}, P)$in $\mathbb{S}_1$, $\mathbb{S}_2$ and $\mathbb{S}_3$. The verifier verifies the correctness of $\mathbb{S}_1, \mathbb{S}_2$ and $\mathbb{S}_3$ based on V_{priv} , V_{sign} and V_{cp} to confirm that it is not blacklisted. Then, it finally maps the hashed data with (R, i, m, q) user's membership private key.

IV. Results

Securing SaaS cloud infrastructure using TPM based provisioning model has been developed for providing service in highly authenticated and secured cloud computing environment. The system model presented has been developed on Visual Studio 2012 framework 4.0 with C#. The overall system has been developed and implemented with Microsoft Windows Azure platform.

We mainly focused on data leakages that can occur in the cloud environment while providing the services. Portable TPM based user attestation architecture supports hardware-based key management by using TPM devices to provide better security and hence device portability is attained. Confidentiality must be attained while accessing the services on the cloud, thus with TPM it provides better security since the accessing of the services can be done only if the TPM exists. Thus, a user can access to cloud storage's contents in secure environment and securely store user data to the remote cloud server using this portable devices which provides added security.The developed system has been simulated on live Microsoft Windows Azure cloud for different performance service parameters like container creation overhead, SaaS cloud memory utilization and the Qos perspective for CPU utilization. The relative study for these all factors has been performed. This system or model performance has been verified for various user size with the assigned authentication devices and the effectiveness as well as performance parameters have been checked for its robustness justification.

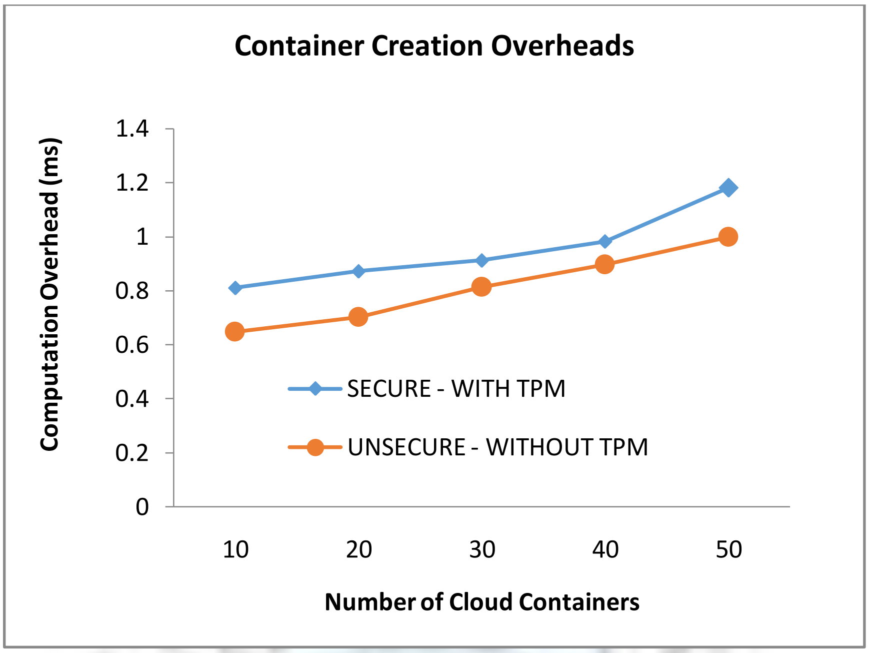


Figure 1. Container creation overhead

Based on the simulated data, the graph (Figure 1) is plotted making the comparison of the overheads in creating the containers of our proposed system with portable TPM device against the overheads in creating the containers without TPM. The computation overheads with and without TPM [17] is being evaluated in milliseconds. Without the external device it is obvious that the computation is of less value. Therefore, from the figure it is evaluated that the average computation overhead without the TPM device (without added security). The average computation overhead with the usage of TPM which provides additional security is evaluated. Thus, the average computational overhead increase is ≈ 17ms which is very negligible when considering a highly secure cloud environment with the portable TPM of cryptographic protocols assuring secured services.

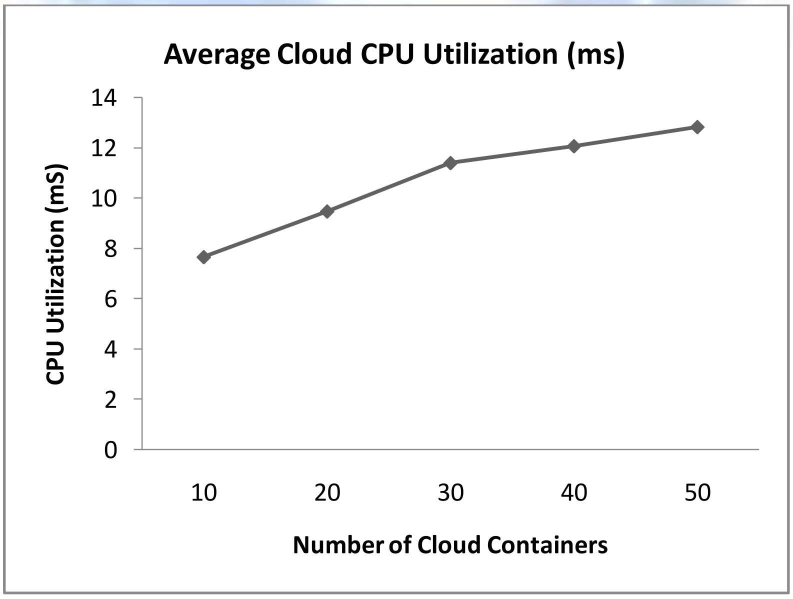


Figure 2. Average Cloud CPU Utilization

There must be the processing time of the virtual machines considered when accessing the cloud services. The average cloud CPU utilization is been depicted in milliseconds which is plotted in the above graph. For every user creating the containers with the cloud services, the CPU is utilized. Here, users are accessing the cloud with the portable TPM devices and the average cloud CPU utilization is plotted. As the service container creations increase from 10 to 50, the processing time also increases. The average utilization of the CPU is found to be ≈ 10.7ms.

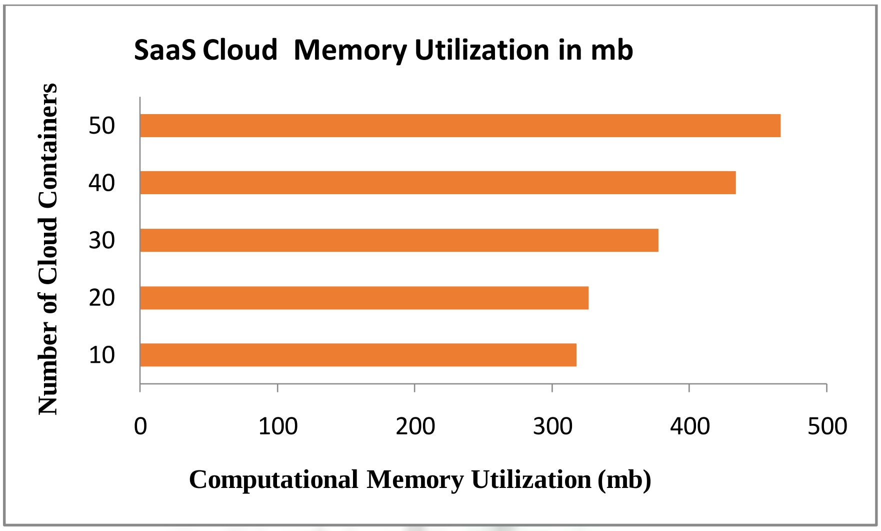


Figure 3. SaaS Cloud Memory Utilization

The above mentioned figure (Figure 3) depicts the cloud memory utilization in megabytes based on the respective set of creating the cloud containers from 10 to 50. Here, the memory utilization is computed based on the user which is able to access the cloud service through his credentials along with the additional authenticated device, TPM. Usually for users to access cloud, cloud providers may be concerned about the memory utilization of varied users. From the graph, it can be justified that not much memory is utilized with the additional security parameter. It clearly shows that even though the cloud containers are 50, the cloud memory utilization is not differing much. Thus, memory computation is highly adaptive. Therefore from these results, we have established that the proposed model can be an effective, secure and optimum adaptable approach for portable TPM based user attestation architecture for cloud environment along with provisioning the services of SaaS.

V. CONCLUSION

Here we have presented the design and implementation of Securing SaaS Cloud Infrastructure using TPM based provisioning, a novel secure service attestation framework for cloud users provisioning storage as a service. One of the main concern in accessing the services from the cloud environment is the security. Privacy and security plays a major role in such environment. Preserving the confidentiality of the data while accessing services and hence, the services should not be misused based on the credentials set by the cloud provider for the particular cloud user. There can be chances that the credentials can be compromised and thus, accessing the SaaS by malicious attackers. We have designed a hardware based device called TPM which is portable for providing better security. With this approach, analysis over consistency and inconsistency attestation methods are done to pinpoint colluding attackers more efficiently than existing techniques. The service provisioning is dynamic in nature. Services are accessed only in existence of TPM. With this service, user can create as many containers as required. The credentials along with TPM are encrypted with a set of security models such as public key cryptographic protocols and carried out a security analysis on our protocol. Hashing technique is used to verify the data from the user which reflects in reduced time constraints. The MVC architecture used in our system is highly efficient. This model acts only based on the request, which means only the services which are needed is run. MVC has route-based URLs i.e., URLs are partitioned into controllers and process and furthermore it is considered on controller not on physical document. Thus reducing the overall computations. The experimental results of our proposed framework shows significant output in terms of CPU utilization, service parameter such as creating containers and the cloud memory overheads due to the existence of MVC architecture. Finally we conclude that our scheme is more secure, efficient and practical than existing schemes.

REFERENCES

[1]. Software as a Service, http://en.wikipedia.org/wiki/Software as a Service, 2013.

[2]. Lomotey, R.K.; Deters, R., "SaaS Authentication Middleware for Mobile Consumers of IaaS Cloud," Services (SERVICES), 2013 IEEE Ninth World Congress on , vol., no., pp.448,455, June 28 2013-July 3 2013

[3]. Tiantian Zhang; Yuliang Shi; Meng Xu; Lizhen Cui, "A Service Provisioning Strategy Based on SPEA2 for SaaS Applications in Cloud," Cloud and Green Computing (CGC), 2012 Second International Conference on , vol., no., pp.290,295, 1-3 Nov. 2012

[4]. Yao Wang; Yaqiang Mao; Yuan Luo, "An In-Out-VM measurement architecture against dynamic attacks in clouds," Communication Technology (ICCT), 2012 IEEE 14th International Conference on, vol., no., pp.761,767, 9-11 Nov. 2012.

[5]. Gartner, Inc. (NYSE: IT) http://www.gartner.com/newsroom/id/2613015.

[6]. Saxena, S.; Sanyal, G.; Srivastava, S., "Mutual authentication protocol using identity-based shared secret key in cloud environments," Recent Advances and Innovations in Engineering (ICRAIE), 2014 , vol., no., pp.1,6, 9-11 May 2014.

[7]. W. Xu, V.N. Venkatakrishnan, R. Sekar, and I.V. Ramakrishnan, “A Framework for Building Privacy-Conscious Composite Web Services,” Proc. IEEE Int’l Conf. Web Services, pp. 655-662, Sept. 2006.

[8]. Zhendong Ma; Manglery, J.; Wagner, C.; Bleier, T., "Enhance Data Privacy in Service Compositions through a Privacy Proxy," Availability, Reliability and Security (ARES), 2011 Sixth International Conference on , vol., no., pp.615,620, 22-26 Aug. 2011.

[9]. Senk, C.; Dotzler, F., "Biometric authentication as a service for enterprise identity management deployment: a data protection perspective," Availability, Reliability and Security(ARES), 2011 Sixth International Conference on , vol., no., pp.43,50, 22-26 Aug. 2011.

[10]. Shuai Han; Jianchuan Xing, "Ensuring data storage security through a novel third party auditor scheme in cloud computing," Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on , vol., no., pp.264,268, 15-17 Sept. 2011.

[11]. Ahmed, M.; Yang Xiang, "Trust Ticket Deployment: A Notion of a Data Owner's Trust in Cloud Computing," Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on , vol., no., pp.111,117, 16-18 Nov. 2011.

[12]. IT Business Edge, c/o QuinStreet, Inc. http://www.itbusinessedge.com/blogs/integration/data-privacy-integration-rank-as-top-saas-concerns-for-large-companies.html.

[13]. Ghosh, N.; Chatterjee, D.; Ghosh, S.K.; Das, S.K., "Securing Loosely-coupled Collaboration in Cloud Environment through Dynamic Detection and Removal of Access Conflicts," Cloud Computing, IEEE Transactions on , vol.PP, no.99, pp.1,1.

[14]. Juan Du; Dean, D.J.; Yongmin Tan; XiaohuiGu; Ting Yu, "Scalable Distributed Service Integrity Attestation for Software-as-a-Service Clouds," Parallel and Distributed Systems, IEEE Transactions on , vol.25, no.3, pp.730,739, March 2014.

[15]. TPM Main Specification, Part 1: Design Principles, ver. 1.2, Trusted Computing Group, 2003.

[16]. M. Burnside and A.D. Keromytis, “F3ildCrypt: End-to-End Protection of Sensitive Information in Web Services,” Proc. 12th Int’l Conf. Information Security (ISC), pp. 491-506, 2009.

[17]. Trusted Computing Group, https://www.trustedcomputing group.org/home, 2013.

[18]. S. Berger et al., “TVDc: Managing Security in the Trusted Virtual Datacenter,” ACM SIGOPS perating Systems Rev., vol. 42, no. 1, pp. 40-47, 2008.

[19]. J. Garay and L. Huelsbergen, “Software Integrity Protection Using Timed Executable Agents,” Proc. ACM Symp. Information, Computer and Comm. Security (ASIACCS), Mar. 2006.

[20]. Tsu-Yang Wu, Tung-Tso Tsai, Yuh-Min Tseng, "Revocable ID-based signature scheme with batch verification", 2012 eight international conference on Intelligent information Hiding and Multimedia Signal Processing, IEEE, 2012,pp 49-54.

[21]. Lianfen Huang , Ying Huang, Zhibin Gao, .lianan Lin, XueyuenJiang,"performance of authentication protocols in LTE Environments ", 2009 international conference on computational intelligence and security, IEEE, 2009 pp 293-297.