Global Journal of Computer Science and Technology: B
Cloud and Distributed
Volume 15 Issue 1 Version 1.0 Year 2015
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Portable Tpm Based User Attestation Architecture for Cloud Environments

By Mr. Pramod & Dr. B R Prasad Babu

Abstract- Cloud computing is causing a major shift in the IT industry. Research indicates that the cloud computing industry segment is substantial and growing enormously. New technologies have been developed, and now there are various ways to virtualize IT systems and to access the needed applications on the Internet, through web based applications. Users, now can access their data any time and at any place with the service provided by the cloud storage. With all these benefits, security is always a concern. Even though the cloud provides accessing the data stored in cloud storage in a flexible and scalable manner, the main challenge it faces is with the security issues. Thus user may think it's not secure since the encryption keys are managed by the software, therefore there is no attestation on the client software integrity. The cloud user who has to deploy in the reliable and secure environment should be confirmed from the Infrastructure as a Service (IaaS) that it has not been corrupted by the mischievous acts. Thus, the user identification which consists user ID and password can also be easily compromised. Apart from the traditional network security solutions, trusted computing technology is combined into more and more aspects of cloud computing environment to guarantee the integrity of platform and provide attestation mechanism for trustworthy services. Thus, enhancing the confidence of the IaaS provider.

Keywords: *TPM, IaaS, vTPM, cTPM, SMRR, SMM,TCG, TED, DRTM, VLR, DRTM, CA.*

GJCST-B Classification : *C.2.1, C.5.3*

Strictly as per the compliance and regulations of:

Portable Tpm Based User Attestation Architecture for Cloud Environments

Mr. Pramod [α] & Dr. B R Prasad Babu [σ]

Abstract- Cloud computing is causing a major shift in the IT industry. Research indicates that the cloud computing industry segment is substantial and growing enormously. New technologies have been developed, and now there are various ways to virtualize IT systems and to access the needed applications on the Internet, through web based applications. Users, now can access their data any time and at any place with the service provided by the cloud storage. With all these benefits, security is always a concern. Even though the cloud provides accessing the data stored in cloud storage in a flexible and scalable manner, the main challenge it faces is with the security issues. Thus user may think it's not secure since the encryption keys are managed by the software, therefore there is no attestation on the client software integrity. The cloud user who has to deploy in the reliable and secure environment should be confirmed from the Infrastructure as a Service (IaaS) that it has not been corrupted by the mischievous acts. Thus, the user identification which consists user ID and password can also be easily compromised. Apart from the traditional network security solutions, trusted computing technology is combined into more and more aspects of cloud computing environment to guarantee the integrity of platform and provide attestation mechanism for trustworthy services. Thus, enhancing the confidence of the IaaS provider. A cryptographic protocol adopted by the Trusted Computing Group enables the remote authentication which preserves the privacy of the user based on the trusted platform. Thus we propose a framework which defines Trusted Platform Module (TPM), a trusted computing group which proves the secure data access control in the cloud storage by providing additional security. In this paper, we define the TPM-based key management, remote client attestation and a secure key share protocol across multiple users. Then we consider some of the challenges with the current TPM based attestation techniques. Thus, proposing a potable TPM which is not embedded into the virtual machines so as to provide the efficiency to the cloud users. Using this approach, security of the user is handled in an efficient way. Finally, we demonstrate the effectiveness and efficiency of the proposed schemes through extensive experimental evaluation on the live Microsoft Windows Azure platform.

Keywords: TPM, IaaS, vTPM, cTPM, SMRR, SMM,TCG, TED, DRTM, VLR, DRTM, CA.

I. Introduction

CLOUD computing is undoubtedly the new era of computing. Industry experts believe that notion of perceiving cloud computing as a new technology trend, is all set to grow. Cost factor is the biggest driver for its expected growth. According to Gartner Inc. Cloud computing is a disruptive phenomenon, with the potential to make IT organizations more responsive than ever. Cloud computing promises economic advantages, speed, agility, flexibility, infinite elasticity and innovation. Cloud computing is an internet-based facility to share technological resources, software and digital information. This technological methodology can save a lot of infrastructure cost and pay-as-you-use model can also be offered through the cloud computing solutions. The above mentioned utilities can help small and mid-sized companies to bring down their operational costs. IDC India lead analyst (software and services research), Kamal Vohra stated, "The most attractive feature of this new technology is the prospect of converting large, upfront capital investments in IT infrastructure into smaller, manageable 'pay-per-use' annuity payments." Recent IDC cloud research shows that spending on public IT cloud services will reach $58.4 billion in 2015 and is expected to be more than $107 billion in 2017. Over the 2013–2017 forecast period, public IT cloud services will have a compound annual growth rate (CAGR) of 23.5%, five times that of the industry overall. Software as a service (SaaS) will remain the largest public IT cloud services category, capturing 59.7% of revenues in 2017. IDC predicts that by 2017, 80%+ of new cloud apps will be hosted on six PaaS platforms. Armonk, N.Y. May 2014 announced businesses across the US have ranked IBM the number 1 cloud computing provider, according to an IDC survey of US market preferences for infrastructure - as - a - service (IaaS). Enterprises ranked Amazon 7th, behind Google (5th) and Microsoft (6th). The rankings are based on responses from more than 400 US-based companies.

Privacy and security is the main concern in the communication over a network. Continuous work to ensure the privacy and confidentiality of the data communication have existed for a long time. Cloud computing is the future but not if security problems persist. The major concern that are being trying to recognize is mainly on the security issues over the cloud computing. However, security and privacy are still cited by many organisations as the top inhibitors of cloud services adoption. FaraziSabhaiet. al. [2] describes the well-known Gartner's seven security issues. The basic security issues such as Data leakage, DoS (Denial of Service) attacks are addressed.

Author α: Research Scholar, VTU, EPCET, Bangalore.
e-mail: pramod741231@gmail.com
Author σ: Head of Department, Computer science and Engineering, SEAIT, Bangalore. e-mail: brprasadbabu@gmail.com

Cloud computing services fall into three major categories- Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software-as-a-Service (SaaS). The software applications which are deployed from the cloud infrastructure provided by the cloud providers are accessed by the Software-as-a-Service (SaaS).The cloud providers manage and control the application so that the user does not need to own the software but rather pay for its use through a web API. Platform as a Service (PaaS) lets the users deploy their applications on the provider's cloud infrastructure using programming languages and tools supported by the provider. Finally, Infrastructure as a Service (IaaS) authorizes the deployment and the execution of an environment fully controlled by the user, typically a Virtual Machine (VM) – on the Cloud resources. Typically, the user should purchase the infrastructure such as software, data resource, server, network accessories in order to operate. But here, the user can directly purchase all these resources as outsourced services from directly from the cloud on "pay-as-you-use" basis. Thus, providing efficiency. Here, we focus on the security aspects of the third category of cloud services, i.e., IaaS platforms and more precisely on confidentiality and integrity issues. The problem arises when the user has to preserve the data confidential on the shared platform. Also, care must be taken that once deployed, the integrity of the environment is not corrupted by the mischievous acts.

A novel approach to protect IaaS platforms that confide on the approach established from the Trusted Computing Group (TCG) whichoffer a secured and reassuring environment with the hardware device called the Trusted Platform Module (TPM). TPM designates both the name of a specification detailing a secure crypto processor as well as the implementation of that specification, often called the TPM chip. TPM asserts the virtue of remote authentication and gets interacted with the symmetric key which can be used for various cryptographic purposes, from the protection of network communications to data encryption. In the IaaS context, it ensures that only the remote resource with which the user is communicating using the TCG protocol can interact with the ciphered data.

Zhidong et. al. [6] address the cloud computing security challenges by proposing a solution called the Trusted Computing Platform (TCP). Trusted cloud computing system is built using TCP as the hardware for cloud computing and it ensures privacy and trust. By design, TPMs offer a hardware root of trust bound to a single, standalone device. TPMs come equipped with encryption keys whose private parts never leave the TPM hardware chip, reducing the possibility those keys may be compromised. Assessing security protocols requires more than showing their robustness against a few use cases. Recent advances in automatic protocol analysis tools [4] allow to scale up the attack complexity against the analyzed protocol and detect design errors.

A TPM is a small tamper proof hardware chip embedded in most recent motherboards. This paper presents TPM with the portability, an extension of the TCG's model which possess an additional secret key to the TPM and shares the secret key with the cloud. Therefore, with this, the cloud can create and share the secret keys of TPM and data over multiple platforms which belongs to a single user.

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

II. Related Work

Much work has been done in concern with security issues in Cloud Computing sector. Let us look into some of the survey which exists. [1] presentsc TPM, an extension of the TPM's design that adds an additional root key to the TPM and shares that root key with the cloud. As a result, the cloud can create and share TPM-protected keys and data across multiple devices owned by one user. Further, the additional key lets the cTPM allocate cloud-backed remote storage so that each TPM can benefit from a trusted real-time clock and high performance, non-volatile storage. This paper shows that cTPM is practical, versatile, and easily applicable to trusted mobile applications. By avoiding a clean-slate redesign, we sidestep the difficult challenge of re-verifying the security properties of a new TPM design. Here it demonstrates cTPM's versatility with two case studies: extending Pasture with additional functionality, and re-implementing TrInc without the need for extra hardware. Re-implementing TrInc without the need for extra hardware again causes with the core security issues.

The paper [3] present a novel secure auditing scheme for cloud computing systems. One major problem with auditing schemes is that they are vulnerable to the transient attack (also known as the timed scrubbing attack). This secure auditing scheme is able to prevent the transient attack via modification of the Linux auditing daemon - audit, which creates attestable logs. This scheme utilizes the System Management Mode (SMM) for integrity checks and the Trusted Platform Module (TPM) chip for attestable security. Specifically, it modifies the auditing daemon protocol such that it records a hash of eachaudit log entry to the TPM's Platform Configuration Register (PCR), which gives an attestable history of every command executed on the cloud server. Different from the existing auditing schemes, this scheme is capable of

preventing the transient attack. It has achieved this by modifying the existing Linux auditing daemon as well as making use of existing software and hardware. This scheme can provide clients with greater assurance and trust in cloud computing services. System with Trusted Platform Module (TPM) [14] provides secure boot via the Core Root of Trust for Measurement as well as secure storage for the log file hashes via the Platform Configuration Registers. The CRTM is anextension of the BIOS which will be initialized first, measure parts of the BIOS block, and then pass control back over to the BIOS. Once the BIOS, boot loader, and OS kernel run and pass control to the OS, the expected configuration by examining the TPM's Platform Configuration Register. The main issue here is, any change to the code between CRTM and the OS running will result in anunseen PCR value. The SMRAM is to be properly setup by the BIOS at boot time and to remain tamper-proof from cache poisoning attacks as in [7]. To prevent these attacks, proper hardware configurations, such as System Management Range Register (SMRR) [9], should be used.

A key technology of cloud computing is virtualization, which can lead to reduce the total cost and increase the application flexibility. However along with the se benefits come added security challenges. The extension of Trusted Computing to virtual environments can provide secure storage and ensure system integrity. In [4], it describes and analyse several existing virtualization of TPM (vTPM) designs: software-based vTPM, hardware-based vTPM, para-virtualized TPM and property-based vTPM and analyse each of their limitations. Concerning about security is an important factor that affect the popularity of cloud computing. Incorporation of trusted computing into virtualized systems should significantly enhance cloud computing system security. In this paper, it briefly reviews the concepts virtualization and trusted computing, and proposal the requirements on a virtual TPM facility. It describes and analyse some existing vTPM designs. Finally, it discusses some open issues of the vTPM, using property-based attestation and secure VMvTPM migration protocols are the key research area sofvTPM in the future.

In [5], it proposes DF Cloud, a secure data access control method of cloud storage services to handle these problems found in the typical cloud storage service Drop box. DF Cloud relies on Trusted Platform Module (TPM) [19] to manage all the encryption keys and define a key sharing protocol among legal users. It assumes that each client is mobile device using ARM Trust Zone[13] technology. The DF Cloud server prototype is implemented using ARM Fast model 7.1 and Open Virtualization software stack for ARM Trust Zone. For DF Cloud client, TPM functions are developed in the secure domain of ARM Trust Zone because most ARM-based mobile devices are not equipped with TPM chip. The DF Cloud framework defines TPM-based secure channel setup, TPM-based key management, remote client attestation, and a secure key share protocol across multiple users/devices. There are several security issues in cloud storage services, among these issues we mainly focused on data leakages that can occur in either client side or server-side. DF Cloud exploit client-side encryption technique, remote attestation for client plat form, and hardware based key management to build a secure access environment. DF Cloud also support secure key sharing protocol across the multiple devices or users. It implemented prototype on ARM Fast model to emulate ARM Cortex-A15 core and Open Virtualization's software stack in environment setup. The performance overhead is quiet high, but if it adopts some optimization techniques such as shared memory between two World, then we can reduce overhead introduced in our current implementation.

TPM is able to provide strong secure storage for sensitive data such as passwords. Although several commercial password managers have used TPM to cache passwords, they are not capable of protecting passwords during verification. This [8] proposes a new TPM-based password caching and verification method called Pwd CaVe. In addition to using TPM in password caching, Pwd CaVe also uses TPM during password verification. In Pwd CaVe, all password-related computations are performed in the TPM. Pwd CaVe guarantees that once a password is cached in the TPM, it will be protected by the TPM through the rest of its lifetime, thus eliminating the possibility that passwords might be attacked in memory. Pwd CaVe eliminates the time that passwords stay in the memory during verification, and therefore keep passwords from attacks in memory. Once a password is cached in the TPM, it will never be released out of the TPM, even in later password verification. Again which proves, the user himself cannot be able to change the password even in emergency situations, in which the password is compromised. Thus, not efficient.

In this [10], it address the issues by incorporating a hardware-based Trusted Platform Module (TPM) mechanism called the Trusted Extension Device (TED) together with the security model and protocol to allow stronger privacy of data compared to software-based security protocols. It demonstrates the concept of using TED for stronger protection and management of cryptographic keys and how the secure data sharing protocol will allow a data owner (e.g., author) to securely store data via untrusted Cloud services. Here, it prevents keys to be stolen by outsiders and dishonest authorised consumers. As part of our future work, this work has to improve the performance of this protocol to the extent that it will be feasible in the real-world scenario. It should also aim to incorporate

larger data sizes. Furthermore, it must extend the current work to incorporate further data sharing control.
In addition to security, most of the hardware that is being shipped today is equipped with the TPM which can be used for realization of trusted platforms. Recently several TPM attestation techniques such as binary attestation and property based attestation techniques have been proposed but there are some fundamental issues that need to be addressed for using these techniques in practice. In [11], it considers an architecture where different services are hosted on the cloud infrastructure by multiple cloud customers (tenants). Then it considers an attacker model that is specific to the cloud and some of the challenges with the current TPM based attestation techniques. In this model, the cloud service provider is used as the Certification Authority (CA) for the tenant virtual machines. The CA only certifies the basic security properties which are the assurance on the traffic originating from the tenant virtual machine and validation of the tenant virtual machine transactions. The components of the CA monitor the interactions of the tenant virtual machine for the certified properties. Since the tenant virtual machines are running on the cloud service provider infrastructure, it is aware of the dynamic changes to the tenant virtual machine. The CA can terminate the ongoing transactions and/or dynamically isolate the tenant virtual machine if there is a variation in the behaviour of the tenant virtual machine from the certified properties. Hence this model is used to address the challenges with the current TPM based attestation techniques and efficiently deal with the attacks in the cloud. This model still need to get extended with the functionality of the CA to certify the behaviour of the tenant virtual machines. Since the Node Controller is aware of the dynamic changes to the tenant virtual machine, it has to ensure that the certified properties are satisfied by the tenant virtual machines.

Group signatures have recently become important for enabling privacy-preserving attestation in projects such as Microsoft's NGSCB effort (formerly Palladium). Revocation is critical to the security of such systems. [15] construct a short group signature scheme that supports Verifier Local Revocation (VLR). In this model, revocation messages are only sent to signature verifiers (as opposed to both signers and verifiers). Consequently there is no need to contact individual signers when some user is revoked. This model is appealing for systems providing attestation capabilities. The signatures are as short as standard RSA signatures with comparable security. Security of our group signature (in the random oracle model) is based on the Strong Diffie Hellman assumption and the Decision Linear assumption in bilinear groups. Here, a precise model for VLR group signatures and discussed its implications. It has described a short group signature scheme where user revocation only requires sending revocation information to signature verifiers, a setup we call verifier-local revocation. Here, the signatures are short: only 141 bytes for a standard security level. They are shorter than group signatures built from the Strong-RSA assumption and are shorter even than BBS short group signatures [8], which do not support verifier-local revocation. There are still a number of open problems related to VLR signatures. Most importantly, is there an efficient VLR group signature scheme where signature verification time is sub-linear in the number of revoked users, without compromising user privacy.

Employs a TPM based method to providea minimum Trusted Code Base (TCB) in [12], which can be used to detect the modification of the kernel. It requires advanced hardware features such as Dynamic Root of Trust Measurement (DRTM) and late launch. The scheme is also directly vulnerable to the scrubbing attack because the measurement target is responsible for invoking the integrity measurement.

To overcome all these issues, we have proposed a portable hardware based security preserving model. Our scheme is different from theirs in that, our scheme offers more revocation capabilities than other schemes, and our scheme is built from the strong public key cryptographic assumptions whereas their scheme is constructed using bilinear maps. Thus, a high performance security model is proposed.

III. Proposed System

Let us consider a case where a cloud provider, cloud users, a blacklisting controller 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. 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 keyKare 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 K and the authenticator asks the cloud provider whether the user can 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 message$s$ 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 asks him 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 approvaltoany verifiers. The verifier cannot identify who is the authenticator in a membership approval operation, thus proving anonymity. Blacklist causes verifiers to reject message assigned 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 link able 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 asignature 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. The public key cryptographic protocol is established as follows.

It is computationally infeasible, on input of a random modulus M and a random element $a \in \mathbb{A}_l^*$ compute values $i > 1$ and q such that $q^i \equiv a(mod\ M)$. In other words, for every probabilistic polynomial-time algorithm R,

$$\mathcal{B}[M \leftarrow \mathcal{K}(1^p), a \in \mathbb{A}_l^*, (q,i) \leftarrow R(M,a) : q^i \equiv a(mod\ M) \wedge 1 < i < M] = \phi(p) \quad (1)$$

where $\mathcal{K}(1^p)$ is an algorithm that generates a public keymodulusand$\phi(p)$ is a negligible function.

Let u be an l_u-bit prime and v is anl_v-bit prime such that$v|u-1$. Let $s \in \mathbb{A}_u^*$ be arandom element of order v. Then, for sufficiently large values of $l_u$and l_v, the distribution $\{(s^x, s^y, s^z)\}$ is computationally indistinguishable from the distribution $\{(s^x, s^y, s^{xy})\}$ where x, y and $z$are random elements from$\mathbb{A}_u$. It can beformally stated as, for every probabilistic polynomial-time algorithm R, the Diffie-Hellman assumption is given by:

$$|\mathrm{B}[R(u,v,s,s^x,s^y,s^{xy}) = 1] - \mathrm{B}[R(u,v,s,s^x,s^y,s^z) = 1]| = \phi(p) \quad (2)$$

Where $\phi(p)$ a negligible function and the probabilities is are taken over the choice of u, v, s according to some generation function $\mathcal{K}(1^p)$ and the random choice of x, y, zin $\mathbb{A}_u$.

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 protocol is constructed by the Camenisch-Lysyanskaya signature scheme, where it has two secret messages m_0 and m_1, and attains the CLsignature (membership of the user)on m_0 and $m_1$from the cloud provider through a secure protocol, and thus the user is verified by the verifier. Here, the authenticator chooses two random l_m-bit secret messages $m_0$and m_1, then interacts with the cloud provider, and inthe end obtains (R, i, q) from the protocol such that $R^i G_0^{m_0} G_1^{m_1} Q^q \equiv A(mod\ M)$. The authenticator will check with verifier that the user is verified and possess the CL-signature on the values of m_0 and m_1. This can be done by values (m_0, m_1, R, i, q) such that $R^i G_0^{m_0} G_1^{m_1} Q^q \equiv A(mod\ M)$ .Let $m = m_0 + m_1 2^{l_m}$ the authenticator also computes $P := \mathcal{D}^m\ mod\ u$where$\mathcal{D}$ is a generator of an algebra group wherecomputing discrete logarithms is infeasible, and proves to the verifier that the exponent m is related tom_0 and m_1. In this protocol, it can choose$\mathcal{D}$: the value of $\mathcal{D}$ can be chosenrandomly by the authenticator, or can be derived from theverifier's name by using an appropriate hash function. If authentication key$\mathbb{K}$ was found comprised and its private key R, i, m_0, m_1, q was exposed, the values m_0 and m_1 are extracted and put on a blacklist. The verifier can then check the public key P in thesignature against this blacklist by comparing it with $\mathcal{D}^{m_0 + m_1 2^{l_m}}$ for all pairsm_0 and m_1 on the black list. In our scheme, there are several types of entities: a cloud provider, cloud users, a blacklisting controller and verifiers. The cloud provider and blacklisting controller could be the same entity or separate entities.

Our scheme builds in concern with the cryptographic protocol scheme and uses the Camenisch-Lysyanskaya signature scheme as underlying building block. To simplify our presentation, we modified the cryptographic protocol scheme in the following ways: 1) each user chooses a single secret m instead of two secrets, and 2) the signature operation is performed solely by the user (along with authentication key $\mathbb{K}$), instead of split by two separate entities (authentication key $\mathbb{K}$ and host in the cryptographic protocol scheme).

In the register phase, a cloud user chooses a secret message m and sends the cloud providera

commit mentto m, i.e., $C := G^m Q^{q'}$ whereq' is a value chosen randomly by the user to blind the m. Also, the usercomputes $P := \mathcal{D}_I^m \bmod u$, where $\mathcal{D}_I$ is a number derivedfrom the cloud provider's basename. The user sends (P, C) to the cloud provider. The provider then issues a membership for the user based on C. The cloud provider chooses a random integer q'' and a random prime i, then computes R such that $R^i C Q^{q''} \equiv A (mod\ M)$, and sends the user (R, i, q''). The cloud provider also proves to the user that he computed R correctly.The CL signature on m is then $R, i, q := q' + q''$. The user'sprivate key is set to be (R, i, m, q).A user can now prove that he is a valid memberby proving that he has a CL signature on the value m.This can be done by values of m, R, i and q such that $R^i G^m Q^q \equiv A\ (mod\ M)$. Also, theuser computes $P := \mathcal{D}^m\ mod\ u$ where $\mathcal{D}$ is a random basepicked up by the user, reveals $\mathcal{D}$ and P, and proves that $\log_{\mathcal{D}} P$ is the same as the one in his private key. The value P serves the purpose of blacklist. Same as in the cryptographic scheme, if a user's private key (R, i, m, q) is compromisedand gets exposed to the public, m is put in the blacklist. The verifier can then check P in the signature against the blacklist by comparing it with $\mathcal{D}^{\hat{m}}$ for all $\hat{m}$ in the blacklist. We refer this type of blacklist as private key-based blacklist and use V_{priv} to denote the blacklist of this type.

This scheme supports two additional blacklist methods, one is signature-based blacklist and the other is cloud provider-based blacklist. In signature-based blacklist, suppose a verifier received a signature from an authenticator and then decided that the authenticator was compromised. The verifier reports the signature to the blacklisting controller who later places $(\mathcal{D}, P)$ of the signatureto the signature-based blacklist, where $\log_{\mathcal{D}} P$ is thesecret of the compromised authenticator. To prove membership, auser with private key (R, i, m, q) now needs not only toprove the (R, i, m, q) such that $R^i C Q^{q''} \equiv A (mod\ M)$ but also to prove that m in his private key isdifferent from $\log_{\mathcal{D}} \hat{P}$ for each $(\hat{\mathcal{D}}, \hat{P})$ pair in the signature-based blacklist. We use V_{sign} to denote the blacklist of this type. In the cloud provider-based blacklist, the provider obtained (P, C) from a user when the user registers and laterdecided to revoke this user from some reason. The cloud provider sends (P, H) to the blacklisting controller who places P to the cloud provider-based blacklist, where $\log_{\mathcal{D}_I} P$ is the secret of the blacklisted user. To prove the membership of the user, a user needs to prove that m in his private keyis different from $\log_{\mathcal{D}_I} \hat{P}$ for each $\hat{P}$ in the cloud provider-based blacklist. We use cloud provider V_{cp} to denote the blacklist of this type.

a) Security

Let us consider the security parameters $r_M r_m r_i r_{i'} r_q r_\theta r_\psi r_\mu r_u$ and r_v where $r_M\ (2048)$ is the size of the public-key modulus, $r_v\ (208)$ isthe size of the m's (user's secret, part of membership privatekey), $r_i\ (576)$ is the size of i's (exponent, part of membership private key), $r_{i'}\ (128)$ is the size of the interval the i''s are chosen from, $r_q\ (2720)$ is the size of the q's (random value, part of membership private key), $r_\theta\ (80)$ is the security parameter controlling the statistical property, $r_\psi\ (256)$ is the output length of the hash function used for Fiat-Shamir heuristic, $r_\mu\ (80)$ is the security parameter needed for the reduction in the proof of security, $r_u\ (1632)$ is the size of the modulus u, and $r_v\ (208)$ is the size of the order v of the subgroup of $\mathbb{A}_u^*$ that is used for blacklist checking. We require that

$$\begin{aligned} r_\theta + r_\psi + 2 + max\{r_m, r_{i'}\} \\ < r_i r_M + r_\theta r_\psi \\ + max\{r_m + r_\mu + 3, r_\theta \\ + 2\} < r_q, \quad r_m = r_q \end{aligned} \tag{3}$$

The parameters r_u and r_v should be chosen such that the discrete logarithm problem in the sub group of $\mathbb{A}_u^*$ of order v with u and v being primes such that $u \in [2^{r_u - 1}, 2^{r_u} - 1]$ and $v \in [2^{r_v - 1}, 2^{r_v} - 1]$, has about the same difficulty as factoring r_M-bit public-key modulus.

b) Generating authentication keys

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 $M = u_M v_M$ with $u_M = 2u'_M + 1, v_M = 2v'_M + 1$ such that u_M, u'_M, v_M, v'_M are all primes, u_M and v_M have the same length, and m has r_M bits.Furthermore, the cloud provider chooses a random generator s' of ZG_M, the group of quadratic residues modulo M. Next, it chooses random integers $e_s, e_t, e_q, e_b, e_g \in [1, u'_M v'_M]$ and computes

$$\begin{aligned} s := s'^{e_s}\ mod\ M; \quad & t := s'^{e_t}\ mod\ M; \\ G := t^{e_s}\ mod\ M; & \\ Q := t^{e_q}\ mod\ M; \quad & A := t^{e_b}\ mod\ M. \end{aligned} \tag{4}$$

It produces a non-interactive proof that s, t, G, Q and A are computed correctly, i.e., $s, t \in \langle s' \rangle$ and $Q, A, G \in \langle t \rangle$. This can be proved using the standard cut-and-choose technique. The cloud provider generates a group of prime order as follows:it chooses random primes u and v such that $u = \mu v + 1$ for some μ with $v | \mu, u \in [2^{r_u - 1}, 2^{r_u} - 1]$, and $v \in [2^{r_v - 1}, 2^{r_v} - 1]$. It then chooses a random $a' \leftarrow \mathbb{A}_u^*$ suchthat $a'^{(u-1)/v} \not\equiv 1 (mod\ u)$ and sets $a := a'^{(u-1)/v}\ mod\ u$. Finally, the cloud provider publishes the public key $(M, s', s, t, G, Q, A, u, v, a)$ and the proof, and stores (u'_M, 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 long term public private key pair (P_I, 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. Analogously, the blacklisting controller has long term public/private key pair(P_M, P_M^{-1}). The blacklisting controller uses its key to sign the blacklist.

c) Verification of the Cloud Provider's Public Key

The user's public key is $(M, s', s, t, G, Q, A, u, v, a)$ and the proof that s, t, Q, A, G are formed properly. Any user inthe system can verify the correctness of the group public key are as follows. Firstly, it verify the proof that$Q, A, G \in \langle t \rangle$and $s, t \in \langle s' \rangle$. Then check whether u and v are primes,$v \mid (u-1), v \nmid \frac{u-1}{v}$and $a^v \equiv 1(mod\ u)$. Later check whether all public key parameters have the required length.

If s, t, Q, A, G are not formed correctly, it couldpotentially mean that the security properties for the usersdo not hold. However, it is sufficient if the users verify theproof that s, t, Q, A, G are computed correctly only once. Also, if a does not generate a subgroup of$\mathbb{A}_u^*$, the cloud provider could potentially use this to link different signatures. As argued in, it is not necessary to prove that M is a productof two safe primes for the anonymity of the users. In fact, itwould be very expensive for the cloud provider to prove that M is a safe-prime product.

d) Registration

This is a protocol which runs between the cloud provider and auser. The public input to this protocol is the user public key $(M, s', s, t, G, Q, A, u, v, a)$ and the cloud provider's long-termpublic key P_I and the cloud provider's basename $name_I$. The privateinput of the cloud provider is the user issuing private key. We assume that the user and the cloud provider have established an authentic channel, i.e., the user needs to make sure that he talks to the right cloud provider and the cloud provider needs to be sure that the user is allowed to register for the membership. Note that we do not require secrecy of the communication channel. Let $\psi(\cdot)$ and $\psi_u(\cdot)$ be two collision-resistant hash functions: $\psi(\cdot) : \{0,1\}^* \longrightarrow \{0,1\}^{r_\psi}$ and $\psi_u : \{0,1\}^* \rightarrow \{0,1\}^{r_u + r_\theta}$. In the register protocol, the user verifies that the user public key $(M, s', s, t, G, Q, A, u, v, a)$is signed by P_I. Then both the user and cloud provider computes $\mathcal{D}_I := \psi_u(name_I)^{(u-1)/v}\ mod\ u$ .The user chooses at random $m \leftarrow \mathbb{A}_v^*$; $q' \leftarrow \{1,1\}^{r_M + r_\theta}$ then computes $P := \mathcal{D}_I^m\ mod\ u$ and $C := G^m Q^{q'}\ mod\ M$. The user sends (P, C) to the cloud provider. Therefore, the user proves to the cloud provider the knowledge of m and q'. He runs as the authenticator of the protocol with the cloud provider as the verifier.

$$\mathbb{a} = \Big((m, q') : C := G^m Q^{q'}\ mod\ u \wedge P := \mathcal{D}_I^m\ mod\ u \wedge m \in \{0,1\}^{r_m + r_\theta + r_\psi + 1} \wedge q' \in \{0,1\}^{r_M + r_\theta + r_\psi + 1}\Big)$$

Thus,

$$QUP\{\mathbb{a}\}(l_I) \tag{5}$$

The cloud provider chooses a random $q'' \leftarrow [2^{r_q - 1}, 2^{r_q} - 1]$ and a random prime$i \leftarrow [2^{r_i}, 2^{r_i} + 2^{r_{i'}}]$ and computes

$$R := \left(\frac{A}{CQ^{q''}}\right)^{1/i}\ mod\ M \tag{6}$$

To convince the user that R was correctly computed,the cloud provider as authenticator runs the protocol

$$QUP\left\{(f) : R \equiv \left(\frac{A}{CQ^{q''}}\right)^f\ mod\ M\right\}(m_C) \tag{7}$$

with the host so that,

a. The user chooses a random integer $m_C \leftarrow \{0,1\}^{r_\psi}$ and sends m_c to the cloud provider.

b. The cloud provider randomly chooses $\mu_i \leftarrow [0, u'_M v'_M]$ and computes

$$\tilde{R} := \left(\frac{A}{CQ^{q''}}\right)^{\mu_i}\ mod\ M \tag{8}$$

$$z' := \psi\big(M \parallel A \parallel Q \parallel C \parallel q'' \parallel A \parallel \tilde{A} \parallel m_C\big) \tag{9}$$

$$b_i := \mu_i + z'/_i\ mod\ u'_M v'_M \tag{10}$$

and sends z', b_i and (R, i, q'') to the user.

c. The user verifies whether i is a prime and lies in$[2^{r_i}, 2^{r_i} + 2^{r_{i'}}]$, computes

$$\hat{R} := R^{-z'}\left(\frac{A}{CQ^{q''}}\right)^{b_i}\ mod\ M \tag{11}$$

and checks whether $z' = \psi\big(M \parallel A \parallel Q \parallel C \parallel q'' \parallel A \parallel \tilde{A} \parallel m_C\big)$.

The user sets$q := q'' + q'$ and stores(R, i, m, q) as itsmembership private key.

Same as in the cryptographic protocol scheme, the cloud provider proves to the user that R was formed correctly, i.e., R lies in $\langle t \rangle$. In above procedure, the cloud provider proves that $R \equiv \left(AC^{-1}Q^{-q''}\right)^f (mod\ M)$ for some value f .Inthesetupprogram, the cloud provider proves that $QGA \in \langle t \rangle$ .Since $C := G^m Q^{q'}\ mod\ M$, the user can conclude that $R \in \langle t \rangle$. Thereason for requiring $R \in \langle t \rangle$ is to assure that later, in the membership approval protocol, R can be statistically hiddenin $\langle t \rangle$. Otherwise, an adversarial cloud provider

could link signatures generated by users whose R does not lie in $\langle t \rangle$. Notethat schemes such as have prevented this byensuring that M is a safe-prime product and then made surethat all elements are members of ZG_M. However, provingthat a modulus is a safe-prime product is rather inefficientand hence the setup of these schemes is not practical asour scheme.

e) Membership Approval Protocol

The membership approval protocol is a protocol run by an authenticator and a verifier. It consists of login and verify. In the login step, the authenticator initializes the interaction with the verifier by sending a request to the verifier. There are three types of blacklist: private-key-based blacklist, signature-based blacklist, and cloud provider-based blacklist. Therefore, the blacklist V contains three sublists, i.e.,$V = \{V_{priv}, V_{sign}, V_{cp}\}$ LetV_{priv} be the blacklist for private-key-based blacklist, in which each element is a value in $\langle a \rangle$. Let V_{sign} be the blacklistforsignature-based blacklist, in which each element is a pairof values in$\langle a \rangle$. Let cloud provider V_{cp} be the blacklist for cloud provider-based blacklist, in which each element is a value in$\langle a \rangle$. The blacklisting controller maintains the blacklist and regularly publishes the newest blacklisttoeveryone in the system, signed using his private key. Thatis, the blacklisting controller publishes $\{V_{priv}\}_{P_G^{-1}}$, $\{V_{sign}\}_{P_G^{-1}}$ and $\{V_{cp}\}_{P_G^{-1}}$.

The verifier first chooses a message$s$ and a nonce $l_q \leftarrow \{0,1\}^{r_\psi}$. The verifier then sends to the authenticator s, l_q, $\{V_{sign}\}_{P_G^{-1}}$ and $\{V_{cp}\}_{P_G^{-1}}$ as the challenge. After the authenticator receives the challenges from the verifier, the authenticator verifies the content of $\{V_{sign}\}_{P_G^{-1}}$ and $\{V_{cp}\}_{P_G^{-1}}$using the blacklisting controller's public key P_G. Let (R, i, m, q) be the authenticator's private key. For each element $(\mathcal{D}_\alpha, P_\alpha)$in $\{V_{sign}\}$, the authenticator checks whether $\mathcal{D}_\alpha^m \not\equiv P_\alpha (mod\ u)$. If there exists some α such that $\mathcal{D}_\alpha^m \not\equiv P_\alpha (mod\ u)$, it means that the authenticator has been blacklisted, the authenticator aborts the membership protocol. Analogously, for each item P_α in V_{cp}, the authenticator checks whether $\mathcal{D}_\alpha^m \not\equiv P_\alpha (mod\ u)$ where $\mathcal{D}_l$ is the base derived from the cloud provider's basename $name_l$. The authenticator quits the membership protocol if the check fails. Note that the authenticator can directly obtain V from the blacklisting controller and checks whether he has been blacklisted. However, it is not required for the authenticator to conduct such operation. Also note that it is the verifier's responsibility to obtain the latest blacklist from the blacklisting controller. If V_{sign} and V_{cp} in the verifier's challenge are not the latest ones, then there is a chance that some blacklisted users may successfully perform membership proof to the verifier without being detected.

i. *Login*

This step is run by the authenticator. The input to this program is the group 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 noncel_q, the signature-based blacklist V_{sign} and the blacklist-based blacklist V_{cp}. The output to this program is a signature $\mathbb{S}$ produced by the authenticator. Firstly, the authenticator picks a random $\mathcal{D} \leftarrow \langle a \rangle$ and two integers $\mathbb{0}, \mu \leftarrow \{0,1\}^{r_M + r_\theta}$ and computes $\mathbb{P}_1 := Rt^{\mathbb{0}}\ mod\ M$, $\mathbb{P}_2 := s^{\mathbb{0}} t^i (s')^\mu\ mod\ M$, $P := \mathcal{D}^m\ mod\ u$

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

$$\begin{aligned} & QUP\{m, q, i, \mathbb{0}, \mu, i\mathbb{0}, ii, i\mu : A \\ & \equiv \mathbb{P}_1{}^i G^m Q^q t^{-i\mathbb{0}} (mod\ M) \wedge \mathbb{P}_2 \\ & \equiv s^{\mathbb{0}} t^i (s')^\mu (mod\ M) \wedge 1 \\ & \equiv \mathbb{P}_2{}^{-i} s^{i\mathbb{0}} t^{ii} (s')^{i\mu} (mod\ M) \wedge P \\ & \equiv \mathcal{D}^m (mod\ u) \wedge m \\ & \in (0,1)^{r_M + r_\theta + r_\psi + 1} \wedge (i - 2^{r_i}) \\ & \in \{0,1\}^{r_{i'} + r_\theta + r_\psi + 1} \{l_q \parallel s\} \end{aligned} \quad (12)$$

with the following steps:

a. The authenticator picks random integers
$\mu_q \leftarrow \{0,1\}^{r_q + r_\theta + r_\psi}$, $\mu_m \leftarrow \{0,1\}^{r_m + r_\theta + r_\psi}$
$\mu_i \leftarrow \{0,1\}^{r_{i'} + r_\theta + r_\psi}$, $\mu_{ii} \leftarrow \{0,1\}^{r_i + r_\theta + r_\psi + 1}$
$\mu_{\mathbb{0}}, \mu_\mu \leftarrow \{0,1\}^{r_M + 2r_\theta + r_\psi}$, $\mu_{i\mathbb{0}}, \mu_{i\mu} \leftarrow \{0,1\} 2^{r_i + r_M + 2r_\theta + r_\psi + 1}$

b. The authenticator computes
$\widetilde{\mathbb{P}_1} := \mathbb{P}_1{}^{\mu_i} G^{\mu_m} Q^{\mu_q} t^{-\mu_{i\mathbb{0}}} (mod\ M)$ $\widetilde{\mathbb{P}_2} := s^{\mu_{\mathbb{0}}} t^{\mu_i} (s')^{\mu_\mu} (mod\ M)$
$\widetilde{\mathbb{P}_3} := \mathbb{P}_2{}^{-\mu_i} s^{\mu_{i\mathbb{0}}} t^{\mu_{ii}} (s')^{\mu_{ii}} (mod\ M)$ $\tilde{P} := \mathcal{D}^{\mu_m}\ mod\ u$

c. The authenticator computes
$z_1 := \psi(M \parallel s' \parallel s \parallel t \parallel G \parallel Q \parallel A \parallel u \parallel v \parallel a \parallel \mathcal{D} \parallel P \parallel \mathbb{P}_1 \parallel \mathbb{P}_2 \parallel \widetilde{\mathbb{P}_1} \parallel \widetilde{\mathbb{P}_2} \parallel \widetilde{\mathbb{P}_3} \parallel \tilde{P} \parallel s \parallel l_q)$

d. The authenticator computes(over the integers)
$b_q := \mu_q + z_1 \cdot q$, $b_m := \mu_m + z_1 \cdot m$,
$b_i := \mu_i + z_1 \cdot (i - 2^{r_i})$, $b_\mu := \mu_\mu + z_1 \cdot \mu$, $b_{\mathbb{0}} := \mu_{\mathbb{0}} + z_1 \cdot \mathbb{0}$,
$b_{i\mathbb{0}} := \mu_{i\mathbb{0}} + z_1 \cdot \mathbb{0} \cdot i$, $b_{ii} := \mu_{ii} + z_1 \cdot i^2$, $b_{i\mu} := \mu_{i\mu} + z_1 \cdot i \cdot \mu$

e. The authenticator sets
$\mathbb{S}_1 := (\mathcal{D}, P, \mathbb{P}_1, \mathbb{P}_2, z_1, b_q, b_m, b_i, b_\mu, b_{\mathbb{0}}, b_{i\mathbb{0}}, b_{ii}, b_{i\mu})$

The authenticator produces a signature of knowledge that his private key has not been blacklisted

in V_{sign} . Let $V_{sign} = \{(\mathcal{D}_1, P_1), \ldots (\mathcal{D}_{m_2}, P_{m_2})\}$. The authenticator computes the signature of knowledge

$$QUP\{(m) : P \equiv \mathcal{D}^m (mod\ u) \wedge P_1 \not\equiv \mathcal{D}_1^m (mod\ u) \wedge \ldots \wedge P_{m2} \not\equiv \mathcal{D}_{m2}^m (mod\ u)\}(l_q \parallel s)$$

with the following steps:

a. The authenticator chooses a random $\mu \leftarrow \mathbb{A}_v$ and computes $\tilde{P} := \mathcal{D}^{\mu}\ mod\ u$.

b. For $\alpha = 1, \ldots m_2$, the authenticator does the following:

i. The authenticator chooses a random $e_\alpha \leftarrow \mathbb{A}_v$.

ii. The authenticator computes

$$C_\alpha := \mathcal{D}_\alpha^{e_\alpha} mod\ u \qquad E_\alpha := P_\alpha^{e_\alpha} mod\ u$$

$$F_\alpha := C_\alpha^{e_\alpha} mod\ u$$

iii. The authenticator chooses a random integer $\mu_\alpha \leftarrow \mathbb{A}_v$

iv. The authenticator computes

$$\tilde{C}_\alpha := \mathcal{D}_\alpha^{\mu_\alpha} mod\ u\, \tilde{E}_\alpha := P_\alpha^{\mu_\alpha} mod\ u\, \tilde{F}_\alpha := C_\alpha^{\mu_\alpha} mod\ u$$

c. The authenticator computes

$$z_2 := \psi(u \parallel v \parallel a \parallel \mathcal{D} \parallel P \parallel \tilde{P} \parallel C_1 \parallel E_1 \parallel F_1 \parallel \tilde{C}_1 \parallel \tilde{E}_1 \parallel \tilde{F}_1 \parallel \cdots \parallel C_{m_2} \parallel E_{m_2} \parallel F_{m_2} \parallel \tilde{C}_{m_2} \parallel \tilde{E}_{m_2} \parallel \tilde{F}_{m_2} \parallel s \parallel V_{sign} \parallel l_q)$$

d. For $\alpha = 1, \ldots m_2$, the authenticator computes $b_\alpha := \alpha_\alpha + z_2 \cdot e_\alpha\ mod\ v$

e. The authenticator computes $b := \alpha + z_2 \cdot m\ mod\ v$.

f. The authenticator sets

$$\mathbb{S}_2 := (\mathcal{D}, P, z_2, b, C_1, E_1, F_1, b_1, \ldots C_{m_2}, E_{m_2}, F_{m_2}, b_{m_2})$$

The authenticator produces a signature of knowledge that his private key has not been blacklisted in cloud provider V_{cp} . Let cloud provider $V_{cp} = \{\mathcal{D}_1, P_1, \ldots P_{m_3}\}$. The authenticator computes the signature of knowledge

$$QUP\{(m) : P \equiv \mathcal{D}^m (mod\ u) \wedge P_1 \not\equiv \mathcal{D}_1^m (mod\ u) \wedge \ldots \wedge P_{m3} \not\equiv \mathcal{D}_l^m (mod\ u)\}(l_q \parallel s)$$

The authenticator outputs the signature $\mathbb{S} := (\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3)$ and sends $\mathbb{S}$ to the verifier.

Observe that in the sign process, the authenticator proves the knowledge of m such that $\mathcal{D}^m \equiv K\ (mod\ u)$ three times, one in each signature of knowledge. We could merge all three signatures of knowledge together such that the authenticator only needs to prove the knowledge of m once, thus couldimprove the performance of membership approvalslightly. When we present the above sign process, we choose to have three separate proof of knowledge protocols to make our protocol easier to read.

ii. Verify

The group public key is $(M, s', s, t, G, Q, A, a, u, v,)$, the message s, the nonce l_q, the corresponding signature $\mathbb{S} := (\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3)$, and the blacklist $V = \{V_{priv}, V_{sign}, V_{cp}\}$. The verifier verifies the signature asfollows:

1. The verifier verifies that s and l_q are the message and the nonce he sent to the authenticator in the challenge step. The verifier also verifies $(\mathcal{D}, P)$in $\mathbb{S}_1$, $\mathbb{S}_2$ and $\mathbb{S}_3$all matches.
2. The verifier verifies the correctness of $\mathbb{S}_1 = (\mathcal{D}, P, \mathbb{P}_1, \mathbb{P}_2, z_1, b_q, b_m, b_i, b_\mu, b_{\mathbb{O}}, b_{i\mathbb{O}}, b_{ii}, b_{i\mu})$ as follows:

i. The verifier computes $b_i' := b_i + z_1 \cdot 2^{r_i}$ and computes

$$\hat{\mathbb{P}}_1 := A^{-z_1}\ \mathbb{P}_1^{b'_i} G^{b_m} Q^{b_q} t^{-b_{i\mathbb{O}}} (mod\ M)$$
$$\hat{\mathbb{P}}_2 := \mathbb{P}_2^{-z_i} s^{b_{\mathbb{O}}} t^{b_i} (s')^{b_\mu} (mod\ M)$$
$$\hat{\mathbb{P}}_3 := \mathbb{P}_2^{-b_i} s^{b_{i\mathbb{O}}} t^{b_{ii}} (s')^{b_{i\mu}} (mod\ M) \qquad \tilde{P} := P^{-z_1} \mathcal{D}^{b_m}\ mod\ u$$

ii. The verifier verifies that

$$\mathcal{D}, P \in \langle a \rangle, \qquad b_m \in \{0,1\}^{r_m + r_\theta + r_\psi + 1}, \qquad b_i \in \{0,1\}^{r_{i'} + r_\theta + r_\psi + 1}$$

iii. The verifier verifies that

$$z_1 := \psi(M \parallel s' \parallel s \parallel t \parallel G \parallel Q \parallel A \parallel u \parallel v \parallel a \parallel \mathcal{D} \parallel P \parallel \mathbb{P}_1 \parallel \mathbb{P}_2 \parallel \hat{\mathbb{P}}_1 \parallel \hat{\mathbb{P}}_2 \parallel \hat{\mathbb{P}}_3 \parallel \tilde{P} \parallel s \parallel l_q)$$

3. The verifier verifies that the authenticator's private key has not been black listed in V_{priv} , where $V_{priv} = \{m_1, \ldots m_{m_1}\}$. . For $\alpha = 1, \ldots m_2$, the verifier verifies that$P \not\equiv \mathcal{D}^{m_\alpha} (mod\ u)$
4. The verifier verifies the correctness of $\mathbb{S}_2 := (\mathcal{D}, P, z_2, b, C_1, E_1, F_1, b_1, \ldots C_{m_2}, E_{m_2}, F_{m_2}, b_{m_2})$ based on $V_{sign} = \{(\mathcal{D}_1, P_1), \ldots (\mathcal{D}_{m_2}, P_{m_2})\}$. It takes the following steps:

a. The verifier computes $\hat{P} \equiv P^{-z_2} \mathcal{D}^b (mod\ u)$

b. For $\alpha = 1, \ldots m_2$, the verifier does the following:

i. The verifier verifies that

$$C_\alpha E_\alpha F_\alpha \in \langle a \rangle, \qquad b_\alpha \in \mathbb{A}_v, \qquad E_\alpha \neq F_\alpha$$

ii. The verifier computes

$$\hat{C}_\alpha := C_\alpha^{-z_1} \mathcal{D}^{b_\alpha}\ mod\ u, \quad \hat{E}_\alpha := E_\alpha^{-z_1} P^{b_\alpha}\ mod\ u, \quad \hat{F}_\alpha := F_\alpha^{-z_1} C^{b_\alpha}\ mod\ u$$

c. The verifier verifies that

$$z_2 := \psi(u \parallel v \parallel a \parallel \mathcal{D} \parallel P \parallel \tilde{P} \parallel C_1 \parallel E_1 \parallel F_1 \parallel \tilde{C}_1 \parallel \tilde{E}_1 \parallel \tilde{F}_1 \parallel \cdots \parallel C_{m_2} \parallel E_{m_2} \parallel F_{m_2} \parallel \tilde{C}_{m_2} \parallel \tilde{E}_{m_2} \parallel \tilde{F}_{m_2} \parallel s \parallel V_{sign} \parallel l_q)$$

5. The verifier verifies the correctness of $\mathbb{S}_3 := (\mathcal{D}, P, z_3, b_e, b_m, C_1, E_1 \ldots C_{m_2}, E_{m_3}, F)$ based on $V_{cp} = \{\mathcal{D}_1, P_1, \ldots P_{m_3}\}$. It takes the following steps:

i. The verifier verifies that

$$C, E \in \langle a \rangle, \qquad b_e, b_m \in \mathbb{A}_v$$

ii. The verifier computes
$\hat{P} := P^{-z_3}\mathcal{D}^{b_m} \bmod u \quad \hat{C} := C^{-z_3}\mathcal{D}^{b_\alpha} \bmod u, \quad \hat{F} := F^{-z_3}C^{b_\alpha} \bmod u$

6. If all the verifications succeed, the verifier outputs succeed, otherwise outputs fail.

iii. Blacklist

There are three sub lists in the blacklist: V_{priv}, V_{sign}, and V_{cp}. Initially, V_{priv} and V_{sign} are set to beempty, and V_{cp} is set to be $\{\mathcal{D}_l\}$, where $\mathcal{D}_l \equiv T_u(name_l)^{u-1/v} \bmod u$ and $name$ is the cloud provider's basename.There are three ways to blacklist a cloud user. Firstly, when a user is compromised and his private key (R, i, m, q) has been exposed (e.g., on the Internet orembedded into some software), the blacklisting controller verifies the correctness of this exposed key by checking $R^i G^m Q^q \equiv A \ (mod\ M)$, then adds $m$to V_{priv}.

Secondly, when a verifier interacts with some compromised authenticator and finds the authenticator suspicious, the verifier reports the authenticator's signature$\mathbb{S} := (\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3)$ alongwith some other physical evidences to the blacklisting controller. After the blacklisting controllerverifies the evidences and correctness of $\mathbb{S}_1$, he adds (P, U) in $\mathbb{S}_1$ to V_{sign}. Then finally, when the cloud provider wants to blacklist a cloud user (e.g., because that user leaves the group), the cloud provider sends (P, C, Θ) to the blacklisting controller, where the (P, C, Θ) tuple was obtained from the to-be- blacklisted user during the register protocol. The blacklisting controller verifies that correctness of $\Theta$and then adds P to cloud provider blacklist V_{cp}.

When the blacklisting controller renounces a user based on the signature of the user, it needs to make sure that the signature is valid. That is, the signature was signed by a group member. This is to prevent a malicious verifier fromadding arbitrary $(\mathcal{D}, P)$ pair toV_{sign}. Similarly, when the blacklisting controller revokes a user based on (P, C, Θ) fromthe cloud provider, he needs to make sure that Θ is a correct signature of knowledge. This is to prevent the (malicious) cloud provider from adding arbitrary P toV_{cp}. Observe that, the cloud provider can always add new members, create new signatures, and later revoke the members that he created by herself. However, even though the malicious cloud provider can choose P of his choice, he has to know $\log_{\mathcal{D}} P$ in order to create a valid signature $\mathbb{S}$ or know $\log_{\mathcal{D}_l} P$ to create a valid Θ.This is a requirement in our security proof. After the blacklisting controller publishes the blacklist $V$and signs using his private key P_G^{-1}, everyone can verifythe authenticity of this blacklist using the blacklisting controller's public key P_G. In practice, we may assume that the blacklisting controller is trusted. Then, the verifiers trust the blacklisting controller to construct the blacklist in a correct manner. In the model where the blacklisting controller is not completely trusted, the blacklisting controller also needs to publish a compromised private key for each item in V_{priv}, a signature for each item in V_{sign}, and a (P, C, Θ) tuple for each element in V_{cp}. The verifiers have to verify the correctness of each element in the blacklist in the same way as the blacklisting controller does. We show that that even if the blacklisting controller or the cloud provider has been corrupted by the adversary, the anonymity of the honest users is still guaranteed.

The initialize and register have the same performance as in the cryptographic protocol scheme. The cost of membership approval protocol has four parts: proof of knowledge of a membership private key, verification that the private key is not in V_{priv}, proof that the private key does not appear in V_{sign},and proof that the private key does not appear inV_{cp}. The first part of the membership approval protocol is the same as the cryptographic protocol scheme and takes constant time for both the authenticator and verifier. The second part is also the same as the cryptographic protocol scheme and takes m_1 modular exponentiations for the verifier, where m_1 is the size of V_{priv}. The third andfourth parts together take about $6m_2 + 2m_3 + z$ modul are xponentiations for both the authenticator and verifier, where m_2 and m_3 are the lengths of V_{sign} and V_{cp}, respectively, and z is a small constant.Observe that the cost of membership approval is linear to the size of the blacklist and could be quite expensive if the blacklist becomes large. There are two possible ways to control the size of the blacklist. First, divide into smaller groups. If the group size is too big, the blacklist may become large as well. One way is to control the size of the blacklist is to have multiple smaller groups. If a group size was 10,000, and at most two percent of the users would get blacklisted, then the blacklist would have at most 200 items. The drawback of this method is that the verifier needs to know which group the authenticator is in, thus, learns more information about the authenticator. It is a trade-off between privacy and performance.

Second, issue a new group if the blacklist grows too big. If the size of the blacklist is above certain threshold (e.g., two percent of the group size), then the cloud provider can do a rekey process as follows: The cloud provider first creates a new group. Then, each user in the old group proves to the cloud provider that he is a legitimate member of the old group and has not been blacklisted, then obtains a new membership private key for the new group.

f) Membership approval for Resource-Constrained Devices

If the authenticator is a resource-constrained device, such as a TPM, a smart card, or a secure coprocessor; it can outsource part of the signing operation to a semi trusted host. Essentially, the signing

operation is split between a computationally weak device (denoted as the principal authenticator) and a resource a bundant but less-trusted host. Observe that if the host does not cooperate, then it is a denial of service. Thus, the host platform is trusted for performing its portion of computation correctly. However, the host is not allowed to learn the private key of the authenticator or to forge a signature without the principal authenticator's involvement. This model is used in the original cryptographic protocol scheme with a concrete security model.

For our scheme, the same technique from can be applied. Let (R, i, m, q) be the principal authenticator's private key. The principal authenticator sends (R, i) to the host but keeps (m, q). The signing operation in the membership approval can be conducted as follows:

1. The principal authenticator picks a random $\mathcal{D} \leftarrow \langle a \rangle$ and computes $P \equiv \mathcal{D}^m \, (mod\ u)$
2. The principal authenticator sends $(\mathcal{D}, P)$ to the host.
3. The host randomly chooses two integers $\mathbb{O}, \mu \leftarrow \{0,1\}^{r_M + r_\theta}$ and computes
   $\mathbb{P}_1 := Rt^{\mathbb{O}} \ mod\ M, \quad \mathbb{P}_2 := s^{\mathbb{O}} t^i (s')^{\mu} \ mod\ M$
4. The principal authenticator and the host jointly produce a signature of knowledge that $\mathbb{P}_1$ and $\mathbb{P}_2$ are commitments to (R, i) and P was computed using the authenticator's secret m. That is, they compute the signature of knowledge

$$\begin{aligned} & QUP\{m, q, i, \mathbb{O}, \mu, i\mathbb{O}, ii, i\mu : A \\ & \equiv \mathbb{P}_1{}^i G^m Q^q t^{-i\mathbb{O}} (mod\ M) \wedge \mathbb{P}_2 \\ & \equiv s^{\mathbb{O}} t^i (s')^{\mu} (mod\ M) \wedge 1 \\ & \equiv \mathbb{P}_2{}^{-i} s^{i\mathbb{O}} t^{ii} (s')^{i\mu} (mod\ M) \wedge P \\ & \equiv \mathcal{D}^m (mod\ u) \wedge m \\ & \in (0,1)^{r_M + r_\theta + r_\psi + 1} \wedge (i - 2^{r_i}) \\ & \in \{0,1\}^{r_{i'} + r_\theta + r_\psi + 1}\}(l_q \parallel s) \end{aligned} \tag{13}$$

With the following steps:

a. The principal authenticator chooses a random integers

$\mu_q \leftarrow \{0,1\}^{r_q + r_\theta + r_\psi}$, $\quad \mu_m \leftarrow \{0,1\}^{r_m + r_\theta + r_\psi}$

And computes

$\widetilde{\mathbb{P}}_{1_{\mathcal{P}}} := G^{\mu_m} Q^{\mu_q} (mod\ M) \quad \tilde{P} := \mathcal{D}^{\mu_m} \ mod\ u$

And sends $\widetilde{\mathbb{P}}_{1_{\mathcal{P}}}$ and $\tilde{P}$ to the host.

b. The host picks random integers

$\mu_i \leftarrow \{0,1\}^{r_{i'} + r_\theta + r_\psi}$, $\quad \mu_{ii} \leftarrow \{0,1\}^{r_i + r_\theta + r_\psi + 1}$

$\mu_{\mathbb{O}}, \mu_\mu \leftarrow \{0,1\}^{r_M + 2r_\theta + r_\psi}$,

$\mu_{i\mathbb{O}}, \mu_{i\mu} \leftarrow \{0,1\} 2^{r_i + r_M + 2r_\theta + r_\psi + 1}$

c. The host computes

$\widetilde{\mathbb{P}}_1 := \widetilde{\mathbb{P}}_{1_{\mathcal{P}}} \mathbb{P}_1{}^{\mu_i} t^{-\mu_{i\mathbb{O}}} (mod\ M)$

$\widetilde{\mathbb{P}}_2 := s^{\mu_{\mathbb{O}}} t^{\mu_i} (s')^{\mu_\mu} (mod\ M)$

$\widetilde{\mathbb{P}}_3 := \mathbb{P}_2{}^{-\mu_i} s^{\mu_{i\mathbb{O}}} t^{\mu_{ii}} (s')^{\mu_{ii}} (mod\ M) \tilde{P} := \mathcal{D}^{\mu_m} \ mod\ u$

d. The host computes

$z_t := \psi(M \parallel s' \parallel s \parallel t \parallel G \parallel Q \parallel A \parallel u \parallel v \parallel a \parallel \mathcal{D} \parallel P \parallel \mathbb{P}_1 \parallel \mathbb{P}_2 \parallel \widetilde{\mathbb{P}_1} \parallel \widetilde{\mathbb{P}_2} \parallel \widetilde{\mathbb{P}_3} \parallel \tilde{P} \parallel l_q)$

and sends z_t to the principal authenticator.

e. The principal authenticator chooses a random $l_{\mathbb{P}} \leftarrow \{0,1\}^{r_\theta}$ and computes

$z_1 := \psi(z_t \parallel l_{\mathbb{P}} \parallel s)$

And sends z_t and $l_{\mathbb{P}}$ to the host

f. The principal authenticator computes (over the integers)

$b_q := \mu_q + z_1 \cdot q, \qquad b_m := \mu_m + z_1 \cdot m$

And sends b_q and b_m to the host

g. The host computes

$b_i := \mu_i + z_1 \cdot (i - 2^{r_i}), \quad b_\mu := \mu_\mu + z_1 \cdot \mu, \quad b_{\mathbb{O}} := \mu_{\mathbb{O}} + z_1 \cdot \mathbb{O},$

$b_{i\mathbb{O}} := \mu_{i\mathbb{O}} + z_1 \cdot \mathbb{O} \cdot i, \quad b_{ii} := \mu_{ii} + z_1 \cdot i^2, \quad b_{i\mu} := \mu_{i\mu} + z_1 \cdot i \cdot \mu$

h. The host sets

$\mathbb{S}_1 = (\mathcal{D}, P, \mathbb{P}_1, \mathbb{P}_2, z_1, l_{\mathbb{P}}, b_q, b_m, b_i, b_\mu, b_{\mathbb{O}}, b_{i\mathbb{O}}, b_{ii}, b_{i\mu})$

5. The principal authenticator produces a signature of knowledge that his private key has not been blacklisted in V_{sign} and V_{vp}, the same as in the sign algorithm.

Note that the verification operation in the membership approval protocol will change slightly to be consistent with the signing operation. More specifically, the verifier now verifies

$z_1 := \psi(\psi(M \parallel s' \parallel s \parallel t \parallel G \parallel Q \parallel A \parallel u \parallel v \parallel a \parallel \mathcal{D} \parallel P \parallel \mathbb{P}_1 \parallel \mathbb{P}_2 \parallel \widetilde{\mathbb{P}_1} \parallel \widetilde{\mathbb{P}_2} \parallel \widetilde{\mathbb{P}_3} \parallel \tilde{P} \parallel l_q) \parallel l_{\mathbb{P}} \parallel s)$

Also note that the steps 3 and 4 cannot be outsourced to the host, because the host does not know the m value. As we shall discuss in the following Section, for implementing our scheme intamper-resistant hardware devices, the blacklists (V_{priv}, V_{sign}, V_{vp}) expect to be very small, asthese blacklists only grow when there are physical attacks on these devices.

g) Using TPM Hardware

We could have the following benefits using the TPM hardware: 1) less computational work for trusted hardware device, 2) portability and 3) more efficient blacklist mechanism. The main design principle is that the host and the hardware jointly perform the membership approval as the authenticator. The host, if corrupted, could break the anonymity of the user but cannot get to know the user's membership private key. Because in any case, the host can pad some identifier to each message sent by the hardware device. Another advantage of using trusted hardware device is to have more efficient blacklist. Thus, a user is blacklisted in the following cases. The user's membership private key was

removed from the trusted hardware device, and was published widely so that everyone knows this compromised private key, it's been blacklisted. When the user's membership private key was extracted from the trusted hardware device by the adversary. The cloud provider suspects that the user's hardware device was compromised, but has not obtained the user' sprivate key. Thus, blacklisted. The user's membership private key was extracted from the hardware device by the adversary. The blacklisting controller suspects that the hardware device was corrupted. The blacklisting controller obtains a signature from the corrupted device but has not obtained the private key becomes blacklisted. The cloud provider blacklists the user for some management reason, e.g., the user's membership expired. The user is blacklisted from transactions, more specifically the user abuses his group privilege and is blacklisted by the blacklisting controller after the user conducted a membership approval.

IV. Experimental Study

The portable TPM based user attestation architecture for cloud environments model has been developed for 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. 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. Therefore, 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 parameters like cloud memory utilization, user attestation overhead 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.

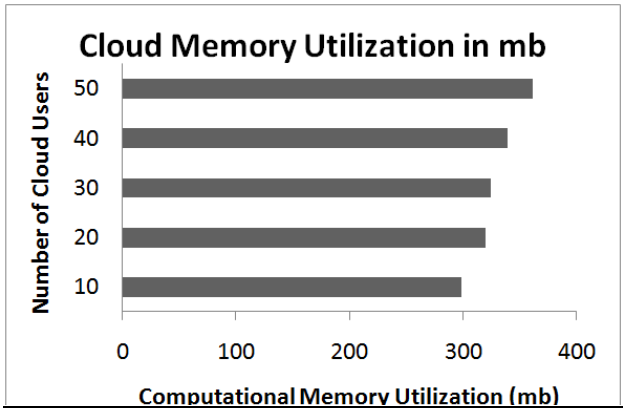

Figure 1 : Cloud Memory Utilization

The above mentioned figure (Figure 1) depicts the cloud memory utilization in megabytes based on the respective set of cloud users 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 users are 50, the cloud memory utilization is not differing much. Thus, memory computation is highly adaptive.

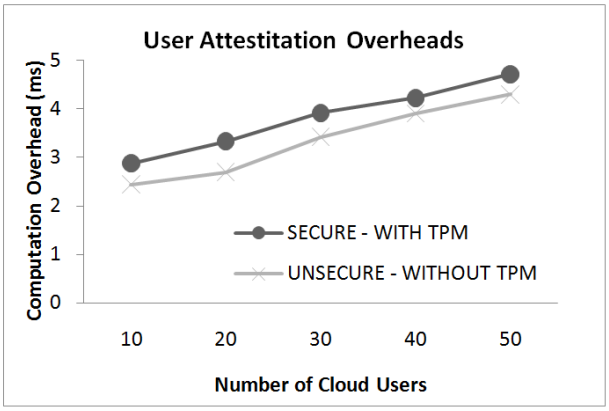

Figure 2 : User Attestation Overhead

Based on the simulated data, the graph (Figure 2) is plotted making the comparison of the user attestation overhead of our proposed system with portable TPM device against the user attestation without TPM. The computation overheads with and without TPM [18] 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) is 5.58ms. The average computation overhead with the usage of TPM which provides additional security is evaluated to be 6.35ms.

Thus, the average computational overhead increase is $\approx 13ms$ which is very negligible when considering a highly secure cloud environment with the cryptographic protocols.

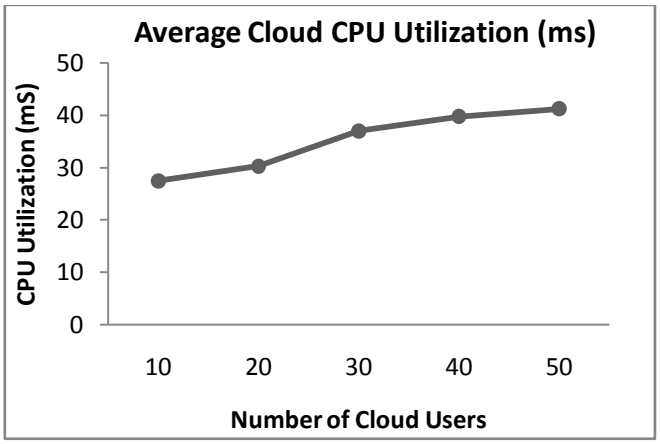

Figure 3 : Average Cloud CPU Utilization

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 interaction 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 users increase from 10 to 50, the processing time also increases. The average utilization of the CPU is found to be $\approx 35ms$.

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.

V. Conclusion

There is a growing demand for sharing data with a large number of consumers using the Cloud. One of the main issues with data sharing in such environments is the privacy and security of information. In particular, the issue of preserving confidentiality of the cloud data and also the need to keep the credentials while respecting the policies set out by the cloud provider. We mainly focused on data leakages that can occur in either client-side or server-side [17]. In this paper we have proposed novel property based attestation techniques for the cloud. We have designed a hardware based device which is portable for further security. We propose a portable device which is used in the authentication and verification of the cloud user. We have discussed our secure data sharing protocol, which allows highly confidential data sharing. The portable TPM based user attestation architecture for cloud environments model exploits client-side authentication with encryption technique to mitigate server-side data leakages such as malicious insider attack or exploiting vulnerabilities of server platform. Due to remote attestation protocol for verifying the client, we ensure that malicious behaviors cannot occur. Therefore, a user can access to cloud storage's contents in secure mobile environment and store user data to the remote server in encrypted form using securely created and managed data encryption key. We also developed a set of security models such as public key cryptographic protocols and carried out a security analysis on our protocol.

Asp.Net MVC is lightweight, provide full control over mark-up and support many features that allow fast & agile development. Hence it is best for developing interactive web application with latest web standards. Thus, our future work we will aim to improve the performance of our protocol based on the Asp.Net MVC Cloud architecture and thus providing security for SaaS cloud with the help of the portable TPM which will be feasible for the cloud users.

VI. Acknowledgment

The authors would like to express their cordial thanks to Mr. Ashutosh Kumar and Mr. Kashyap Dhruve of Planet-/ Technologies for their much valued support and advice.

References Références Referencias

1. Benoit Bertholon, Sebastien Varrette and Pascal Bouvry, "CERTICLOUD: a Novel TPM-based Approach to Ensure Cloud IaaS Security" 2011
2. FarzadSabahi, "Cloud Computing Security Threats and Responses", IEEE 3488 rd International Conference on Communication software and Networks(ICCSN), 27-29 May 2011, pp 245-249, Print ISBN: 978-1-61284-485-5, DOI: 10.1109/ICCSN.2011.6014715.
3. Houlihan, R.; Xiaojiang Du, "An effective auditing scheme for cloud computing," Global Communications Conference (GLOBECOM), 2012 IEEE , vol., no., pp.1599,1604, 3-7 Dec. 2012
4. Xin Wan; Zhiting Xiao; Yi Ren, "Building Trust into Cloud Computing Using Virtualization of TPM," Multimedia Information Networking and Security (MINES), 2012 Fourth International Conference on , vol., no., pp.59,63, 2-4 Nov. 2012
5. Jaebok Shin; Yungu Kim; Wooram Park; Chanik Park, "DFCloud: A TPM-based secure data access control method of cloud storage in mobile devices," Cloud Computing Technology and Science (Cloud Com), 2012 IEEE 4th International Conference on, vol., no., pp.551,556, 3-6 Dec. 2012
6. Zhidong Shen, Qiang Tong " The Security of Cloud Computing System enabled by Trusted Computing Technology", 2 International Conference on Signal Processing Systems, Dalian, (ICSPS), 5-7 July 2010,

Vol 2, pp 11-15, Print ISBN: 978-1-4244-6892-8, DOI: 10.1109/ICSPS.2010.5555234.
7. Duflot "Getting into the SMRAM: SMM reloaded" Proc. of the 10thCanSecWest conference, 2009.
8. Hua Wang; Yao Guo; Xia Zhao; Xiangqun Chen, "Keep Passwords Away from Memory: Password Caching and Verification Using TPM," Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on , vol., no., pp.755,762, 25-28 March 2008.
9. I.Corporation. Software developer's manual vol. 3: System programming guide, June 2009.
10. Thilakanathan, Danan; Chen, Shiping; Nepal, Surya; Calvo, Rafael A.; Liu, Dongxi; Zic, John, "Secure Multiparty Data Sharing in the Cloud Using Hardware-Based TPM Devices," Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on , vol., no., pp.224,231, June 27 2014-July 2 2014
11. Varadharajan, V.; Tupakula, U., "TREASURE: Trust Enhanced Security for Cloud Environments," Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on , vol., no., pp.145,152, 25-27 June 2012
12. J. McCune, B. Parno, A. Perrig, M. Reiter, and H. Isozaki. "Flicker: an execution infrastructure for TCB minimization." Proc. of the ACM European Conference on Computer Systems (EuroSys), March, April 2008.
13. ARM, "ARM Securtiy Technology, Building a Secure System using Trust Zone Technology", 2009
14. Trusted Computing Group. TPM specifications version 1.2. https:llwww.trustedcomputinggroup.org/downloads/specifications/tpm, July 2005
15. Dan Boneh, Hovav Shacham "Group Signatures with Verfier-Local Revocation", Proceeding of the 11th ACM conference on Computer and communications security, NY 2004. Pages 168-177
16. TPM. http://www.trustedcomputinggroup.org/resources/tpm_main_specification
17. Amazon S3, "Using Data Encryption" http://docs.amazonwebservices.com/AmazonS3/latest/dev/Using Encryption.html
18. R. A. Popa, J. R. Lorch, D. Molnar, H. J. Wang, and L. Zhuang. Enabling Security in Cloud Storage SLAs with Cloud Proof. In Proceeding of the 2011 USENIX Annual Technical Conference, June 2011.