TPM Based Design for Enhanced Trust in SaaS Services

Mustapha Hedabou [1,2] **, Ali Azougaghe** [3] **, Ahmed Bentajer** [4]

[1] School of Computer and Communication Sciences
University Mohammed VI Polytechnic, Benguerir. Morocco

[2]Dept. Computer Science. ENSA de Safi
University Cadi Ayyad, Marrakch. Morocco

[3] ENSIAS Mohammed V University in Rabat, Morocco

[4] ENSA School of Tetouan
Abdelmalek Essaadi University, Morocco

Abstract. On the past decade, Trusted Platform Modules (TPM) have become a valuable tool for providing a high level of trust on locally executing software. Indeed, in addition to its availability on most commodity computers, TPM are totally free of cost unlike other available Hardware-Based devices while they offer the same level of security. Enhancing trust in SaaS services regarding the security and the privacy of the hosted SaaS application services can turn out to be a pertinent application scope of TMP. In this paper we present a design for a trusted SaaS model that gives cloud users more confidence into SaaS services by leveraging TPM functionalities combined with a trusted source code certifying authority facility. In our design, the cloud computing provider hosting the SaaS services acts as a root of trust by providing final cloud users insurance on the integrity of the SaaS application service running on its platform. A new mechanism of multisignature is developed for computing a join signature of SaaS service software by the trusted authority and TPM. A prototype implementation of the proposed design shows that the integrity of SaaS application service before and after it was launched on a cloud provider platform is guaranteed at low cost.

Keywords. Cloud computing, SaaS services, TPM, trust, Code source certification, Mutlisignature schemes.

1 Introduction

Cloud computing services demand is booming because they can reduce the cost and complexity of owning and managing computers and networks. Customers have no investment in information technology infrastructure, purchase hardware, or buy software licences because these charges are covered by cloud providers. On other hand, they can

David C. Wyld et al. (Eds): ITCSE, NLCA, ICAIT, CAIML, ICDIPV, CRYPIS, WiMo - 2020
pp. 217-226, 2020. CS & IT - CSCP 2020 DOI: 10.5121/csit.2020.100520

benefit from rapid return on investment, rapid deployment, and customization. Moreover, specialized cloud providers in a particular area (such as e-mail) can bring advanced services that a single company can not provide. Cloud computing is often considered efficient because it allows organizations to devote their resources only to innovation and product development.

Nowadays, cloud services are widely used by businesses. More than 3 millions businesses have adopted Google Apps, Googles's cloud e-mail, calendar and collaboration solutions for businesses. This rate is expanding at 3000 users a month [19].

Despite its potential benefits, many chef executives and IT manager are not willing to lost the control of their data by offloading it to cloud computing platforms. Their main concerns are about the confidentiality and privacy of their data. Their apprehension is amply justified because cloud provider employees can either accidentally or intentionally tamper with the hosted data. This violation can take place without the knowledge of the data's owner.

Many efforts have been done by cloud services providers in order to Strengthen the trust of users by reducing the threat of the insider attacks. For example, they protect and restrict access to the hardware facilities, adopt strict accountability and auditing procedures, and reducing the number of staff who have access to critical components of the infrastructure [16]. Nevertheless, cloud providers employees with administrator privileges can still have access to cloud users data and tamper with it.

To establish more confidence in their services, cloud providers have defined the specification for the widely implemented Trusted Platform Module (TPM) [18, 11]. This approach has been used by Santos et al. [16] to design a trusted cloud computing platform based on TPM attestation chains. In [10], the authors have proposed Terra, a trusted platform based on a virtual machine. Terra has the ability to prevent insider abusers from tampering with the users data. With the capability to provide a remote attestation capability, Terra enable a cloud users to determine upfront whether the host can securely run the computation on their data.

Another approach based on a trusted trusted party acting as a root of trust, that attestes cloud hosted services to their clients, was proposed in [6]. Cloud provider which acts as trusted party, responsible runs programs upon request on behalf users, and issues digital certificates of program's identity (e.g., a code hash) for use by other entities that interact with the running program instance. Unlike the other approach, this model does not use the TPM for issuing the remote attestations.

This paper proposes a new trusted SaaS platform that aims to addresses the trust concerns unique to the SaaS model. Our model allows to reinforce the confidence of the cloud users in both cloud providers and services providers. For this purpose, the source code of the SaaS application service must be certified by a trusted code certifying

authority in order to prevent the cloud provider to tamper with the application service. By using a new multisignature mechanism, the cloud provider trough the TPM, certifies also the source code. This will prevent the missuse of service application when it was running in the cloud platform.

2 Background

The SaaS model is based on the distribured application architecture. It includes components to facilitate and improve the business model. Unlike the traditional software vendor that is only concerned with application functionalities, the SaaS provider is also responsible for operating and managing the environment that supports all their cloud users. This is done by adding more distriburion tiers to support the need of services requests to be routed among more than physical operating environment.

SaaS services are located in high layers, which means in the software level, offer a complete online applications that can be used by cloud users without any development. SaaS enables cloud users to utilise an application on a pay-as-you-go basis without need to install nor to update the application. Maintenance an upgrade are carried out by the service provider as a part of service. The application can be accessed trough web browser or thin client over internet. Early examples of SaaS can be traced back to Hotmail, the e-mail service owned by Microsoft.

The location of the SaaS model in the higher layers increases the difficulty of guaranteeing the security of hosted data because the cloud and services providers have the full control over the software and operating environment that manipulate the cloud users data. Thus, the cloud users data may succumb to insiders attacks from two sides. Cloud provider employees with root privileges can execute any attacks. On other hand, sysadmins of the service provider can launche a malicious or faulty applications services in order to misappropriate cloud users data or to conduct additional harmful functions without the user approval. Consequently, the security of the SaaS must be strengthened more than any of other available delivery models.

In addition to the standard security practices shared with other application delivery models, including firewalls, IPS, IDS, .., SaaS displays more security defenses related to identity management, data storage and data transmission [15]. However, the deployment of all theses countermeasures does not prevent the insiders attacks. To strength the trust of users on their services, cloud providers have proposed hardware and software approaches to enable the construction of trusted platforms. In particular, the Trusted Computing Group (TCG) introduced a standard for the design of the trusted platform module (TPM) [18] chip that is now endowed with commodity hardware.

2.1 Trusted Plaform Module (TPM)

The TPM is a security specification defined by the TCG. It is almost installed on the motherboard of a computer or laptop, and communicates with the rest of the system using a hardware bus through well defined commands. The TPM provides cryptographic operations such encryption, decryption and signing as well as random number generation. It also provides space for the storage of small amount of information such as cryptographic keys. Since it was implemented carefully in the hardware, the TPM is resistant to software attacks [9]. Many researches [2, 10, 17, 16] have proposed to use the features of TPM chips for reinforcing trust in cloud computing platforms.

Each TPM is associated with a number of signing keys. The endorsement private key (EK) identifies the TPM and thus, the physical host. The EK stands for the validity of TPM [18]. The respective manufacturers sign the corresponding public key to guarantee the correctness of the chip and validity of the key. Related to the EK are Attestation Identity Keys (AIKs). An AIK is created by the TPM and linked to the local platform through a certificate for that AIK. This certificate is created and signed by a certificate authority (CA) [9]. In particular, a privacy CA allows a platform to present different AIKs to different remote parties, so that it is impossible for these parties to determine that the AIKs are coming from the same platform. In our model, the used AIK key represents the pulic key of the TPM with regard to multisignature schemes.

The TPM Attestation consists of several steps of cryptographic authentication by which the specification for each layer of the platform is checked from the hardware up to the operating system and application code. At a high level, the TPM attests the source code of service application by signing its hash with an attestation identity key (AIK). This will be done by following the trust chain TPM $\rightarrow$ BIOS $\rightarrow$ BootLoader $\rightarrow$ OS $\rightarrow$ Application [13]. Direct Anonymous Attestation (DDA) [5] can be used to protect the privacy of the TPM in such a way that a user will able to verify the validity of attestation without linking it with the platform that contains the TPM.

2.2 Multignature schemes

Multisignature schemes [12, 14] are designated to enable a group of signers to produce a compact, joint signature on a common message. Any other user can verify the authenticity of a given message based only on the multisignature and all signers public keys. Consider entities $1, \cdots, N$ each having a public key and corresponding secret key. A multisignature (MS) scheme allows, at any time, any subset $L \in 1, \cdots, n$ of users to engage interactively a protocol that outputs is a joint signature on a given message m. Verification can be done by any user given just L, m, the computed multisignature σ , and the public keys of all signers in L. Multisignatures can be useful for contract signing, co-signing, or distribution of a certificate authority.

Earlier implementations of a multisignature σ on a message m is obtained by setting $(\sigma_i : i \in L)$ where σ_i is is signature on the message m. This multisignature is however large, in particular of size proportional to the number |L| of signers. Moreover, Earlier multisignature schemes require a set up process between all the signers which make their use impraticable especially for devices with small ressources such as PDAs, cell phones and TPM chips. Research efforts have lead to recent multisignature schemes [3, 4] that require nothing more than that each signer has a certified public key. Furthermore, the these multisignature schemes have become as efficient as others signature schemes in both signing and verification process.

In 2003, a non interactive multisignature scheme based on the signature of Boneh, Lynn and Shacham [8] was proposed by Boldyreva [7]. Let G be a Diffie-Hellman group of prime order p and let g be a generator of G. Let H be a description of a random member of the family of hash functions and $\{P = P_1, \cdots, P_n\}$ be the group of players. Any player $P_j \in P$ with a secret key $sK_j = x_j$, that wishes to participate in signing takes M, computes and broadcasts $\sigma_j \leftarrow H(M)^{x_j}$. Let $L = \{P_1, \cdots, P_l\}$ be a subgroup of players participating to the signing process. Let $J = \{1, \cdots, l\}$ denotes the indices of involved players. The leader, which can be any player, collects the signature of each player and computes $\sigma = \prod_{j \in J}(\sigma_j)$ and outputs $T = (M, L, \sigma)$.

The verifier takes $T = (M, L, \sigma)$ and the list of public keys of the players involved in signing $L = \{pK_1, \cdots, pK_l\}$ where $pK_i = g^{sK_i}$ for each $i \in L$. The verifier computes $pK_l = \prod_{j \in J}(pK_j) = \prod_{j \in J}(g^{sK_j})$ and outputs $V_{DDH}(g, pK_l, H(M), \sigma)$. We recall that $V_{DDH}(g, u, v, h, \sigma)$ outputs valid if $log(u)_g = log(h)_v$ and false otherwise.

3 Proposed trusted SaaS design

In a SaaS model, the service provider launches the application service as an instance hosted on a physical platform owned by a cloud provider. cloud users gain access to the service application through an API supplied by the cloud provider. The TPM facilities are used to provide cloud users with evidence of well defined security properties of the platform that hosts the application service. This only security practice deployed in the cloud platform does not provide cloud users with any evidence about authenticity of the application service running in the platform which can lead to lack of trust in SaaS services.

The proposed trusted SaaS design aims to mitigate trust issue on SaaS sevices by ensuring the integrity of a SaaS service application before and after it wa running on a cloud provider's platform. For this purpose, the service provider requests a trusted authority in order to certify the source code of the service application, Prior to deliver it to the cloud provider. The cloud provider certifies also the source code of the application service trough the TPM technology before to launch it. The signatures of

the trusted certifying authority and the cloud provider are sealed to each other via the multisignature schemes to produce a compact, joint signature on the source code of the application service .

By signing a source code of an application service, a trusted authority certifies the Authenticity and integrity of the source code. In some ways, the trusted authority binds the proprieties verified by the application to the signature of its source code. In our trusted SaaS model, the service provider requests a trusted certifying authority to sign the source code of its service application before to host it in a cloud computing platform. Digital signatures contain proof of content integrity so that the source code cannot be altered which gives users a serious base to determine the trustworthy in the service application and its behavior. Once this step achieved, the service provider forwards the service application and certificate issued by the trusted authority to the cloud provider for attesting its code source by using the TPM facilities.

3.1 Multisignature scheme using TPM

Even if TPM 2.0 products and systems have important security advantages over TPM 1.0, including elliptic curves digital signature ECDSA, it still not support signature based on decisional and computational Diffie-Hellman problems. Therefore, the multi signature scheme prposed by Boldyreva [7] can not be implemented directly by TPM. In this paper, we propose to use Diffie-Hellman oracle process introduced by Acar et al. [1] to modify Boldyreva's scheme to meet the RSA-based signature requirements of the TPM.

In [1], the authors use TPMv2 API commands with Schnorr signature in order to define a function $f_x : G \rightarrow g$ such that $f_x(h) = h^x$, where x is the TPM Diffie-Hellman private key. By exploiting the Diffie-Hellman oracle process, we can compute $H(M)^x$ for a message M by applying the function f_x to $H(M)$.

In our design only two players are involved in multisignature process, namely the certifying code trusted authority and the TPM. Let P_1 denotes the trusted authority and P_2 the TPM. By using the same notations as in section 2.2, let $sK_1 = x_1$ and $pK_1 = g^{x_1}$ the couple of private and public keys of trusted authority P_1. The trusted authority computes the signature $\sigma_1 \leftarrow H(M)^{x_1}$ of a message M and make it available to other players. A TPM with a private key $sK_2 = x_2$ compute the signature M $\sigma_2 \leftarrow H(M)^{x_2}$ by applying the Diffie-Hellman oracle process to $H(M)$. The TPM signature is broadcasted to other players. The leader collects the signature of each player, computes $\sigma = \prod_{j \in \{1,2\}}(\sigma_j)$ and outputs $T = (M, L, \sigma)$. In our design, the dealer is the SaaS service application provider.

To verify the validity of the multisignature $T = (M, L, \sigma)$, the final cloud user takes $T = (M, L, \sigma)$ and public keys of the trusted authority and the TPM ($pK_1 = g^{x_1}$,

$pK_2 = g^{x_2}$), computes $pK = pK_1 \times pK_2 = g^{x_1+x_2}$ and outputs $V_{DDH}(g, pK, H(M), \sigma)$.

3.2 Prototype implementation

In this section we describe the prototype implementation of the proposed trusted SaaS platform. We describe the protocols and commands used for certifying the source code of the SaaS application service by the cloud provider and the trusted authority. The private and public keys of the trusted authority and TPM are a simple keypair of an asymmetric cryptographic scheme, namely Diffie-Hellman keys. Figure 1 depicts the architecture of the proposed design

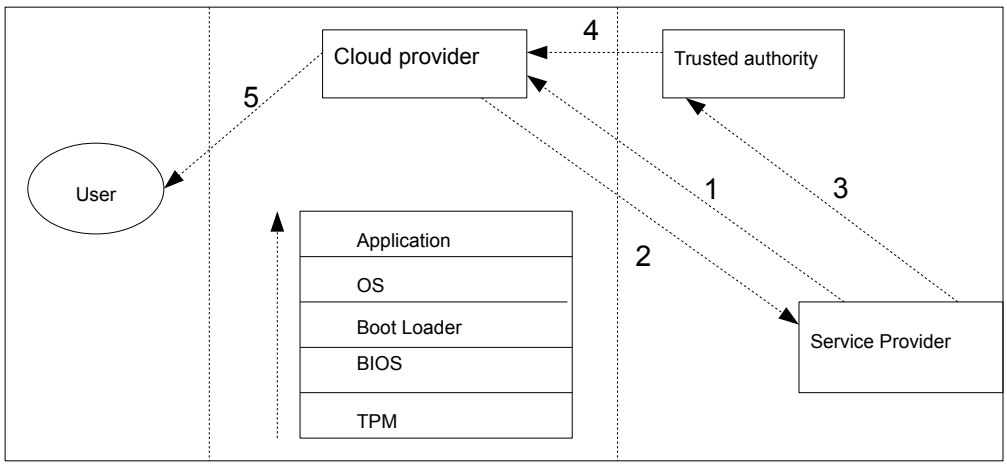

FIGURE 1 – Architecture of the proposed trusted SaaS platform

In our implementation, we used an auto-signed code source certificate through Openssl. Other more trusted authorities that support Diffie-Hellman based signature, such as Certum, can be leveraged for issuing Code Signing Certificate. In order to compute the TPM signature, we have issued the TPMv2 API command ***TPM CreatePrimary*** for generating the private/public key (x_2, g^{x_2}) of TPM. The call of ***TPM Commit*** with

input h(M) where M is the formated code source of the service application outputs $W = H(M)^w$ for a random $r \in Z$ and the command ***TPM Sign*** with an arbitrary input c outputs $r = cx + w$. We derived the signature of the TPM by computing $(H(M/W))^{1/c} = H(M)^{x_1}$ wrapped in a $TPMU\_SIGNATURE$ structure. In order to compute the multisignature issued by the trusted authority and the TPM, we have extracted the relevant bits from the file.bin that contains the TPM Signature.

3.3 Security study

The code source certification of the application service by the a trusted authority and the cloud provider addresses the specific need to prevent the misuse of the application before and after it was launched. The use of multisignature schemas establishes a mechanism of double protection since it prevents attackers from service and cloud providers to act jointly or separately in order to misrepresent the behavior of the application . The attestation provided by the trusted authority ensures the integrity of the application when it was under the monitoring of the cloud provider. After launching the application, its control is under the service provider but the cloud provider still control the server platform and the launch procedure. Thus, the cloud provider can be responsible for guaranteing the application's integrity after it was launched by using the TPM certification. In various TPM-based platform, such as Intel's Trusted Technologies [6], the proposed functionalities have been extended to post-launch checks. This makes the task of the cloud provider more affordable.

Trust in a cloud provider is mainly based on its reputation, therefore it has no interest to corrupt the behavior of the service instance it hosts. It can strength assurances of trustworthiness to its cloud users by issuing attestations of its own software stack based on a hardware root of trust. The cloud provider is also responsible for granting the privileges access the admin interface to the service provider. Thus, it can limit the control of the service provider on the application after it was launched to only the legitimate operations (launch, stop, ...). For all these reasons, the cloud provider itself can be seen as a trusted platform that run a well tested software stack and offers hosting platform for multi tenants users. This means that the cloud provider in our model can serve as a guarantor for preventing the service provider from tampering with application service after it was launched even if it was still under its control.

The mere fact that the cloud provider acts as a root of trust in our model does not mean that it is completely trusted. since it can still tamper with application service before running it. This issue was addressed by incorporating a trusted certifying source code authority that ensures the application's integrity when it was under the control of the cloud provider. This process allows to prevent any attempt for subverting the instance before it was hosted in the cloud platform.

4 conclusion and Future Work

In this paper we have proposed a new trusted SaaS model that mitigates the trust issues in SaaS delivering model by reinforcing the cloud users confidence in SaaS application services. In our model, the source code of the SaaS service application is certified by a trusted authority and the cloud provider via TPM by using a new multisignature mechanism that we have developed for this purpose. The proposed SaaS model gives the cloud users the ability to cheek the integrity of the SaaS service application before and after it was running in a cloud platform. In addition, we have implemented a prototype of the proposed design in a local environment. In the future, we plane to implement an instance of our design for deployment in real cloud computing environments.

Références

[1] T. Acar L. Nguyen and G. Zaverucha. *A TPM Diffie-Hellman Oracle.* Cryptology ePrint Archive : Report 2013/667, 2013.

[2] S. Berger, R. Caceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn. *vTPM : virtualizing the trusted platform module.* In Proc. of USENIX-SS'06, Berkeley, CA, USA, 2006.

[3] M. Bellare, G. Neven. *New multi-signatures and a general forking lemma.* in CCS06, 2006.

[4] M. Bellare, G. Neven. *Identity-based multi-signatures from RSA.* In CT-RSA, 2007.

[5] E. Brickell, J. Camenisch, L. Chen. *Direct Anonymous Attestation.* In ACM Conference on Computer and Communications Security, pp. 132-145, 2004.

[6] A. Brown, J. S. Chase . *Trusted Platform-as-a-Service : A Foundation for Trustworthy Cloud-Hosted Applications.* In : Proc. of CCSW. pp. 15-20 (2011).

[7] A. Boldyreva *Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme.* in International Workshop on Theory and Practice in Public Key Cryptography (PKC) 2003 Proceedings, LNCS Vol. 2567, pp. 31-46.

[8] D. Boneh, B. Lynn and H. Shacham *signatures from the Weil pairing.* In Asiacrypt 01, 2001.

[9] *Common Criteria.*Trusted Computing Group (TCG) Personal Computer (PC) Specific Trusted Building Block (TBB) Protection Profile and TCG PC Specific TBB With Maintenance Protection Profile, July 2004.

[10] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. *Terra : A Virtual Machine-Based Platform for Trusted Computing.* In Proc. of SOSP 03, 2003.

[11] M. Hedabou. *Cryptography for Addressing Cloud Computing Security, Privacy, and Trust Issues.* In Computer and Cyber Security : Principles, Algorithm, Applications, and Perspectives, CRC Press, 2018.

[12] K. Itakura and K. Nakamura,. *A public-key cryptosystem suitable for digital multisignatures.* In NEC Res. Development 71 (1983), pp. 1-8.

[13] C. Nie. *Dynamic root of trust in trusted computing.* In TKK T1105290 Seminar on Network Security, 2007.

[14] T. Okamoto. *A digital multisignature scheme using bijective public-key cryptosystems.* Commun. ACM Trans. Computer Systems 6, 8 (1988), 432-441.

[15] *SaaS Security and privacy.* http ://www.progress.com/en-gb/docs/whitepapers/public/SaaS/SaaS-Security.pdf.

[16] N. Santos, K. P. Gummadi, and R. Rodrigues. *Towards trusted cloud computing.* In Proceedings of the Workshop on Hot Topics in Cloud Computing, HotCloud'09. USENIX Association, 2009.

[17] R. Sailer, T. Jaeger, E. Valdez, R. Caceres, R. Perez, S. Berger, J. L. Griffin, and L. v. Doorn. *Building a MAC-Based Security Architecture for the Xen Open-Source Hypervisor.* In Proc. of ACSAC'05, Washington, DC, USA, 2005.

[18] *Trusted platform module.* http ://www.trustedcomputinggroup.org/developers/trusted platformmodule.

[19] Ron Zalkind. *Protecting Your Data In Google Docs Compliance In The Cloud.* http ://www.cloudlock.com/pdf/Protecting-Your-Data-In-Google-Docs.pdf.