Accountability in Cloud Computing by Means of Chain of Trust

Dipen Contractor, and Dhiren Patel
(Corresponding author: Dipen Contractor)
Computer Engineering Department, NIT Surat, India
(contractor.dipen@gmail.com, dhiren29p@gmail.com)
(Received Nov. 17, 2015; revised and accepted Feb. 11 & Mar. 6, 2016)

Abstract

Cloud computing offers various services in form of infrastructure, platform, and software to meet the consumer requirements. It is radically changing how information technology services are created, delivered, accessed, and managed. However, this swift has prompted concerns regarding security and privacy due to cloud computing characteristics such as the multi-tenancy, elasticity, and layered architecture. One of the major challenge is to offer accountability in cloud services across all the dependencies. When one entity relies on other entities for functioning, it creates a dependency in system and makes it difficult to sort out the responsible entity among them. In this paper, we analyze the problem of creating accountable cloud services. We utilize basic functionality provided by Trusted Computing Group (TCG) in form of chain of trust (CoT) by securely recording identities (of entities). We propose a solution that modifies existing chain of trust to build accountable cloud computing. We explore dependency relationship in building reliable chain of trust in cloud and define it for better implementation.

Keywords: Accountability, chain of trust, cloud computing, dependency

1 Introduction

Cloud computing is an amalgamation of technologies like service oriented architecture (SOA) and virtualization, that turns Internet into service delivery infrastructure. Service providers can lease a set of resources from cloud infrastructures to provide their software as services in an economical way without owning physical infrastructures [3].

Various Cloud service models serve as forms of abstraction and eliminate the need to deal with internal details of the operation, management, and state of the underlying infrastructure [27]. The cloud service provider's (CSP's) computing resources are pooled to serve all consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned according to consumer demand. The customer generally has no control of the location of the allocated resources. As a result, establishing accountability in distributed and layered architecture is an issue. The problem arises when you consider that the application is dependent on functioning of entities in order to continue processing, and thus a single entity failure could stop the entire application.

According to Cloud Computing Incidents Database (CCID), major CSPs have suffered downtime ranging from a few minutes to a few hours [10, 30]. During a cloud service disruption, affected customers will not be able to access the cloud service and in some cases may suffer degraded performance. For example, in June 2013, major cyber-attack was launched on North Korea by South Korea [20, 23]. The attack compromised an update of application hosted at cloud service provider that also hosts North Korean government websites. Many government websites were defaced by this attack. According to Trend Micro [24], website defacement was only the tip of the iceberg; sensitive information (of military and government) was also compromised.

Trusted computing architecture [26] offers a concept of a chain of trust (CoT). We explore this concept using few additional operations to model dependencies in distributed and layered architecture of cloud. Accountability of chain is rooted from tamper resistant hardware and identity (and integrity) of each component running on a particular platform can be assured. Propagation of chain follows the principle of measure before loading [26]. It means that the entity that is executed; measures the identity of the next entity (to be executed), then passes on the execution to the measured entity. This functionality is associated with SElinux, which provides isolation of execution of any program with LSM (Linux security module) hooks [11]. According to David et al. [14], any mechanism that promotes accountability should have following two basic features:

1.1 Tamper-resistance

A mechanism to promote accountability should deter and detect any modification or malfunctioning in it [6]. Mechanism must be tamper resistant or at least tamper evident. No entity can bypass assessment operations, and if it tries, it can be identified easily. Consumer and provider both can rely on such mechanism and present it as a proof to any third party (if dispute arises). In fault detection, one can decide responsibility and act accordingly.

1.2 Privacy and Transparency Balance

Accountability promotes control and transparency in system [15]. Keeping record of all entities brings transparency, however; this may help the attacker to launch specific attack or leads rivals to know insights of cloud. Maintaining privacy while recording identities will equalize transparency.

Considering dependence relationships in cloud computing with features mentioned above, we present formalization of Chain of Trust applicable with service level agreement or third party certificates.

Rest of this paper is organized as follows: Section 2 discusses cloud computing basics. In Section 3, we discuss dependencies in cloud scenario. In Section 4, we discuss chain of trust and its formation in cloud computing. In Section 5, we present formal representation of chain of trust with conclusion and references at the end.

2 Background

National Institute of Standards and Technology (NIST) has defined the Cloud Computing as [12].

A model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

2.1 Service Delivery Models

The Service model describes an organization's scope and control over the computing environment and characterizes the level of abstraction for its use. As shown in Figure 1, three well-known and often used service models are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

Cloud computing is not restricted to Infrastructure/Platform/Software as a Service; it can be further extended to provide a variety of service models. Armbrust et al. [4] coin the phrase "X as a Service (XaaS)"; where X can be anything like data, management, security etc. that can be provided to consumer as a service.

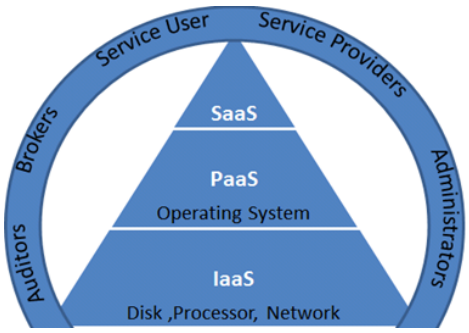

Figure 1: Cloud computing with layers

2.2 Cloud Actors

The NIST cloud computing reference architecture recognizes the main actors in a cloud ecosystem, their activities and functions in terms of cloud computing.

Service Consumer: A service consumer is a person or an organization that uses service from, one or more cloud service providers.

Cloud service provider (CSP): A cloud service provider is an organization, or entity responsible for making a cloud service available to interested parties.

Service provider (SP): A service providers is an organization, or entity responsible for building or combining individual services such as IaaS, PaaS, or SaaS.

Cloud auditor: A cloud auditor is a party that can perform independent assessment of services, system operations, performance, and security of the cloud implementation.

Cloud broker: A cloud broker is an entity that manages the use, and delivery of cloud services, and/or negotiates relationships between cloud service providers and cloud consumers.

Cloud carrier: A cloud carrier is an intermediary (actor) that provides connectivity of cloud services from providers to cloud consumers.

3 Dependencies

Cloud Computing environment consists of a number of players (actors) that interact in fragile manner, to benefits for their own and for others. Individual service providers can independently manage policies, and controls cloud entities. Cloud computing ecosystems enable highly dynamic and effective organizational collaborations. Organizations (dispersed geographically) can provide services from different levels of abstraction (e.g. business, architecture, or programming). These abstractions create dependencies.

Many researchers have found different types of dependencies in cloud computing [16, 28], such as *organizational*

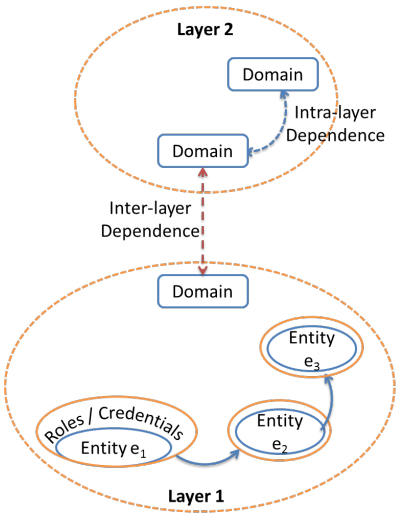

Figure 2: Conceptual model for architectural dependencies in cloud computing

and *architectural*. Organizational dependencies further classified (based on their existence) as *inter-layer* and *intra-layer* dependencies. According to Siani Pearson et al. [16] organizational dependencies may arise in situation where cloud service is composed from different services provided by different service providers. Due to that, accountability of system is shared with a cloud provider as well as with individual service providers.

Consumers must understand the scope of system management and monitoring, including access management, change management, configuration management, patch management, and vulnerability management of individual service providers.

3.1 Architectural Dependencies

We conceptualize the architecture of cloud computing to elaborate architectural dependencies. We try to intricate entity, service, domains, and layer for understanding dependence relationship.

As depicted in Figure 2, Domain represents a grouping of similar entities inside a layer. E.g. infrastructure (IaaS) security domain includes physical access control mechanism for physical resources. Each domain contains a predefined policy [21] based on that; rules, credentials or attributes are assigned to each entity. An Entity can be defined as physical resource (e.g. memory or disk), a process, or services in cloud computing. Functionality of an entity depends on other entities as shown in Figure 2. Entities from different domain communicate through layers. Entities from different domains and layers can be accounted in a single chain. The dependence relation is the relation that exits between entities of different domains across layers. Each domain operates with different policy so it is essential to handle dependence relation carefully in chain construction.

3.2 Organizational Dependencies

This section intends to support a discussion of accountability aspects of cloud computing by presenting simple usage scenarios from client's perspective. Depending on the deployment model (i.e. private, community, public, and hybrid); cloud providers, and users interact differently. Traditionally client server architecture does not have dependency relationships but in a cloud like environment, it could be between layers (SaaS, PaaS and IaaS). Evolving public cloud services are complex and dependent on providers and provider to provider as connections. In fact, the SaaS service you receive may be provided by another IaaS provider [13].

To elaborate the situation, we present highly outsourced scenario of cloud computing. We assume a cloud service provider borrows platform from PaaS provider to host applications of different software providers. The PaaS provider might have leased infrastructure from public or private IaaS provider. As shown in Figure 3, the situation is similar to hybrid cloud computing. The main issue lies in the form of establishing accountability. For example; if a cloud service consumer complains about malfunctioning of a particular service, then how cloud provider will come to know which layer or domain has a fault?. Assuming that cloud service provider has identified a particular service provider; Since ownership of the infrastructure belongs to another service provider, it is difficult for cloud service provider to investigate without proof. The solution to this problem lies in securely keeping identities of all the entities involved from different layers or providers.

4 Accountability in Cloud Computing

Accountability is about defining governance to comply in a responsible manner with internal and external criteria, ensuring implementation of appropriate actions, explaining and justifying those actions and remedying any failure to act properly [15].

Accountability and its different attributes for distributed dependable systems are briefly studied by Siani Pearson [16] and other researchers. They have worked in A4cloud project [1] for promoting accountability in cloud architectures. Accountability could be divided in two types; prospective accountability (preventive controls) and retrospective accountability (detective controls). Detective controls for the cloud include secure and trustworthy auditing, tracking, reporting, and monitoring of system.

Main components of this accountability notion are transparency, responsibility, provision for assurance, and satisfaction of obligations [14].

For the purposes of this paper, accountable cloud maintains a tamper-evident record that provides non-repudiable evidence. Based on this record, a faulty node

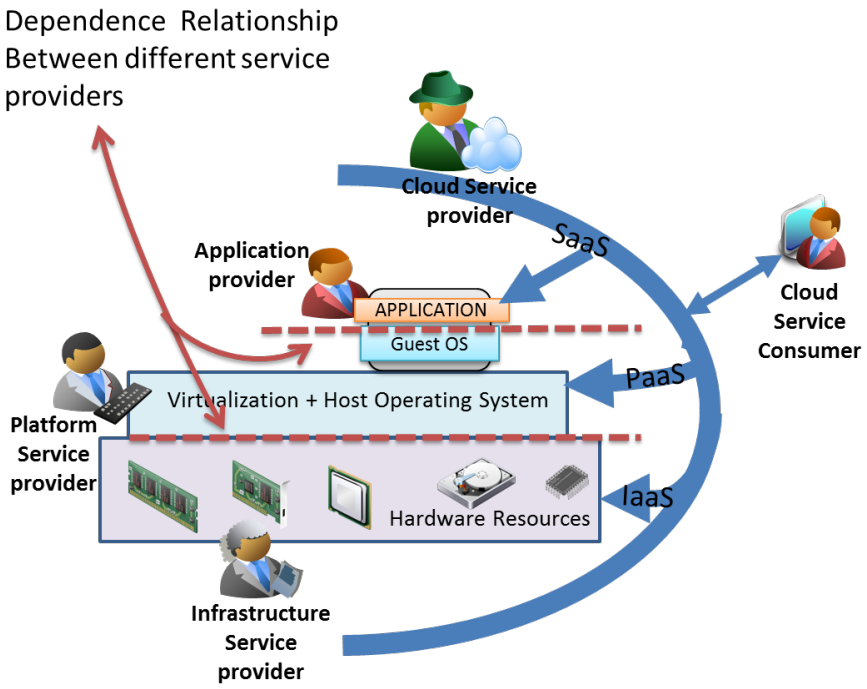

Figure 3: Cloud computing services outsourced from different providers

(whose observable behavior deviates from that of a correct node) can be detected. Accountable cloud provides primitives for cloud carriers to validate the identities of entities associated in cloud.

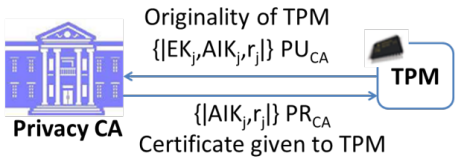

Figure 4: Certificate issued to TPM from privacy CA

We setup a chain of trust that could be fully embedded in to each layer. This implies that the consumer needs to know all information regarding identities of entities involved in service orchestration [29]. To provide tamper evidence feature, we also maintain original hash of individual modules as they load to identify mismatched entity. Individual service providers are accountable for their own layer activity. Hence, consumers should understand the dependency of their application on all services and assess risks pertaining to third-party service providers. CSPs have been reluctant to share information relating to platform security using the argument that it could provide insights to hackers. However, consumers should demand transparency from CSPs and seek information necessary to perform risk assessment and ongoing security management. Before utilizing services from any CSP, consumer can ask about all identities of all the entities and service providers involved in that services.

5 Chain of Trust

As explained earlier, we need to keep track of all the entities that are involved in service creation at each layer. A chain of trust is a term used to describe the sequence of hashes that incorporates different entities that spawns over multiple layers in a cloud [19].

The first element of the chain (Root) should be reliable and it can vouch for its accountability (e.g. IBM's 4758 secure processor [8] and a tamper-evident hardware chip [22]). During the initialization of platform, Root entity is loaded first, and then other modules are loaded. The Root records identity (in terms of hash) of the second element after booting of the platform and continue to build the chain. The second element then records identity of the third element in the chain within the layer. As the second element is already assessed, it assesses the third element's integrity, and so on. These hashes will be securely sent to consumer (or trusted third party) for verification. With availability of actual hash from original manufacturer and reference database [25], one can easily identify mismatched entity.

Chain of trust concept technically relies on TCG (Trusted Computing Group) architecture for recording hash and reporting of it using cryptographic primitives. To this effect, TCG specifies a hardware module, the Trusted Platform Module (TPM) [18]. TPM is a tamper resistant piece of cryptographic hardware built onto the system board. It implements primitive cryptographic functions, using which more complex features can be derived.

The manufacturer embeds a unique master keypair

named as endorsement key (EK) during the creation of TPM chip. The private part of EK never leaves out of the chip. It also embeds mechanism [2] for a certificate on the public key-part of EK, which vouches for the authenticity of TPM. This certificate allows a third party or consumer to verify that messages signed with this EK come from a genuine TPM. Moreover, it allows a third party to communicate over a trusted channel with the TPM. However, due to privacy concerns, the EK is not used directly but an intermediary attestation identity key (AIK) is used, which is wrapped in a certificate, signed by an externally trusted certificate authority (CA) (as shown in Figure 4).

Verification of CoT requires certificate given by any trusted third party; in our case Privacy CA. This certificate assures involvement of accountable service provider. Moreover, a certificate $Certi_j(AIK_j, r_j)$ also contains hash of root entity r_j, so that it can be verified. Storing hash of entities will require large and secure storage. TPM comes with a limited number of registers named platform configuration registers (PCRs). Also at application layer, two lists are maintained viz, stored measurement list (SML) and integrity measurement list (IML) [9]. To permit a TPM version to perform in the cloud, specifications have been generated for a virtual TPM (vTPM) [17] that provides software instances of TPMs for each virtual machine. As shown in Figure 7, a chain is built across layers with a single root (as in a private cloud deployment environment). In other case, highly outsourced cloud would have individual root for each layer, so we name it as multi-rooted chains. Working of CoT is explained with few basic operations as discussed below, which combines various functionalities of TPM.

5.1 Extend Operation

As defined in reference architecture of TPM [26], extend operation maintains the final single hash of a platform. It discards individual hashes after adding to a single value.. Verifier has to derive all the steps and try to get that single hash value. As explained in previous section, hashes of individual identities are sequentially stored. For identification and verification of dependencies in CoT, we have changed traditional extend operation as shown in Figure 5. CoT also maintains original hash of the modules of entities as evidence and provided when explicitly asked by the consumer or third party. Extend operation stores actual hash and extended hash of individual entities to different list. For operating on PCRs, only TPM can write or extend it. SML and IML are utilized for storing hashes encrypted with AIK at application level and sent whenever they are required.

CoT comprises identities of entities and actors involved in cloud service life cycle with privacy. Let us define entity set $E = \{e_1, e_2, e_3, e_4, \ldots, e_n\}$. Hash of the each entity has one to one mapping to set $H = \{h_1, h_2, h_3, \ldots, h_n\}$. Each entity belongs to a particular domain from set $D = \{d_1, d_2, d_3, \ldots, d_n\}$. Each domain operates at a particular layer of set $L = \{l_1, l_2, l_3, \ldots, l_n\}$. Mainly cloud standard architecture is based on SPI (software, platform, infrastructure) framework [31] and that contains only three layers. To define dependence relationship, we need to define dependency in cloud. We begin with platform configuration registers. Root represents a first element in chain whose hash r1 is recorded in initial register.

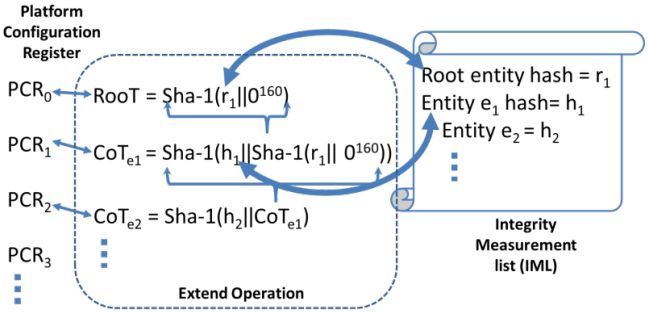

Figure 5: Extend operation in CoT: Our proposal

$$PCR0 = Root = SHA1(r_1 \parallel 0^{160}).$$

Subsequently;

$$\begin{aligned} PCR1 &= CoT_{e1} = SHA1(h_1 \parallel SHA1(r_1 \parallel 0^{160})) \\ PCR2 &= CoT_{e2} \\ &= SHA1(h_2 \parallel SHA1(h_1 \parallel SHA1(r_1 \parallel 0)) \\ &= SHA1(h_2 \parallel CoT_{e1}). \end{aligned}$$

5.2 Dependency Operation

Dependency is expressed by both extending and hashing, symbolized by dependency operator Π. The functionality of entity e_2 is dependent on e_1, and it can be represented as

$$Dep_{e2} = (e_2 \Pi e_1) = SHA1(h_2 \parallel CoT_{e1}).$$

In similar way, individual domain's CoT can be formalized as

$$CoT_{d1} = Dep_{en} = (e_n \Pi e_{n-1} \Pi e_{n-2} \Pi \ \cdots \Pi e_1).$$

As described earlier, multiple domains are contained in a layer, so layer's dependency in a private cloud scenario (single rooted chain) can be shown as

$$CoT_{l1} = Dep_{dn} = (d_n \Pi d_{n-1} \Pi d_{n-2} \Pi \cdots \Pi d_1).$$

5.2.1 Dependency Relation

As explained in architectural dependency section, dependency relation exits between two different layers (or domains) which may be owned by different service providers and therefore there exits multi-rooted CoT.

Typically, for cloud service provider, it can be represented as,

$$\begin{aligned} CoT_{csp} &= Dep_{ln} \\ &= (l_n \Pi l_{n-1} \Pi l_{n-2} \Pi \ldots \Pi l_1) \\ &= (l_{SaaS} \Pi l_{PaaS} \Pi l_{IaaS}). \end{aligned}$$

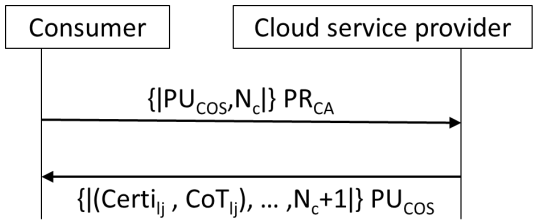

Figure 6: Transfer of CoT

Cloud service provider sends an individual layer wise CoT with its certificate which is used for authenticity and verifiability. Formally, certificate of a layer can be $Certi_{lj} = AIK_j, r_j$ where j is a number of that layer.

Here, AIK proves identity of a service provider and r (hash of the root) can be used for verification of CoT while maintaining privacy. The extend operation preserves the order of dependency; an entity cannot pretend to occur after a certain event as ordering is automatic. Numbers of PCRs are limited on TPM chip so SML will be utilized afterward. Hashing reduces the amount of data that needs to be stored, and extended in order to detect manipulation.

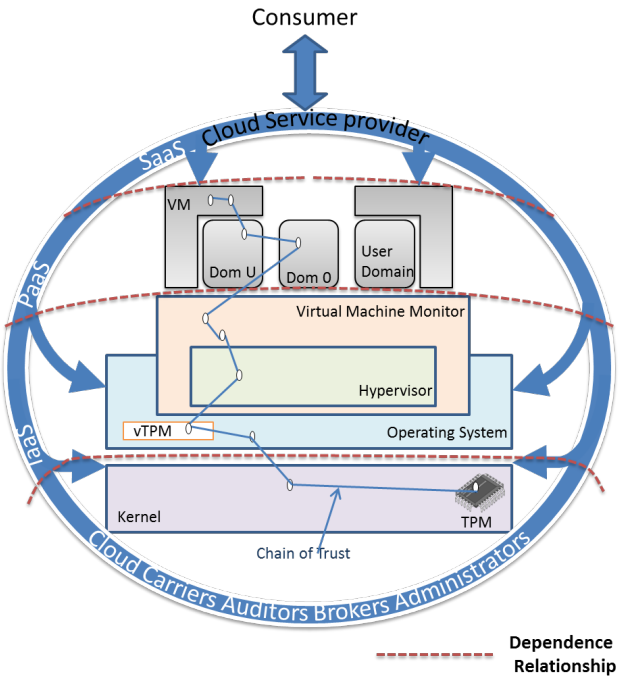

Figure 7: Cloud computing and CoT

5.2.2 Verification of CoT

Verification of CoT will be done at consumer side, but it may be delegated to third party based on computational powers. TPM works well with asymmetric key cryptography; while keeping in mind adversary present on the network. Initially, consumer sends a certificate containing public key PU_{COS} and nonce N_c (Nonce is used to ensure freshness of certain responses), given by trusted third party CA. Then CSP will reply all individual CoT of layers (denoted by j) with its certificates, encrypted with consumer's public key.

Our formal model creates a single chain of trust that can accommodate different roots and handle dependence relationships. Verification of this chain can be done by trusted third party or even at consumer side (with the help of reference manifest database) [25]. These reference hashes are collected from the original source: i.e. the software and hardware manufacturers. Each certificate provides identity of a service provider. After matching all the hashes of CoT, Root hash notifies the completeness of the chain to consumer.

5.2.3 Implementation of CoT in Cloud Computing

In our experiments; we use host machine with Ubuntu 12.04 and Xen 4.3.0 hypervisor [5] based cloud test bed with various domains. Domain-0 is the highest privileged domain; consumer operates at individual domain-U. Without enabling TC (TPM chip from BIOS), we initiated domain-0, then we compiled a user kernel and from which we created our Master domain for TC. We then enabled TC from BIOS. We kept minimal functionality and less interfaces for this domain. Now, from this domain, we can initiate individual domain CoT. Each domain-U receives a vTPM instance for integrity measurement. Consumer can ask for complete CoT (from infrastructure entity resources to SaaS resources).Therefore, this CoT approach is useful to both parties viz; CSP and consumers. CSP can rectify a fault and decide responsibility, and end-user can present it as a proof for remediation [15].Currently we have implemented it using basic scripting language i.e. python. From our previous work [7], we utilized communication mechanism to get individual chain from different domains. Actual PCR values and its corresponding CoT values are shown in Figure 8.

6 Conclusion

Everything as a service concept of cloud environment allows easier utilization of resources of different providers but it makes difficult to establish accountability of low-level entities. We propose chain of trust (CoT) as one solution to provide recording, transferring, and verifying identities of entities. Offering transparency while maintaining privacy is achieved with CoT. Verifying individual terms will lead to tamper evidence property of CoT. Secure generation of keys and certificate denote tamper resistance nature of system and thus CoT could be an acceptable solution to manage and verify architectural and organization dependencies present in cloud computing.

Extend Operation

No.	Component's Hash value	Component name
c1	298df125b260ef64201bdf0815c003873eedd50e	[BIOS:EV_S_CRTM_VERSION]
c2	B794d4c2eddf03b69bdc8ac0a1cbf9278fe131ca0	[BIOS:EV_POST_CODE(EV_CODE_NOCERT)]
c3	878e8aae5975ed177335c2cc60685dbeec04af17	[BIOS:EV_EFI_PLATFORM_FIRMWARE_BLOB]
c4	9069ca78e7450a285173431b3e52c5c252990473	[BIOS:EV_SEPARATOR, 00000000]
c5	4e0209993c236720356703f4f66b7cb4439710cb	[BIOS:EV_S_CRTM_CONTENTS]
c6	060b13c687d713e9fa2d0d0a8ef51a05ade90a2c	[BIOS:EV_S_CRTM_CONTENTS]

PCR_0	**66 3F 8E E7 5A 5C 33 FD EE 81 AF 20 FA 9E 7F 65 1F 47 91 35**

Similarly , We received Following values

PCR_1	**BB C0 45 C1 9F 21 F5 4F 49 B1 38 95 EC 49 A9 EC BF 15 F8 1D**
PCR_2	**0C E8 06 1E BA 5C 53 2C 62 59 5D DB 30 75 CA E8 A8 57 53 61**
PCR_3	**B2 A8 3B 0E BF 2F 83 74 29 9A 5B 2B DF C3 1E A9 55 AD 72 36**
PCR_4	**51 D5 3B C6 51 32 A5 76 55 20 7B 5D B3 24 42 7C A6 47 3D FA**
PCR_5	**45 A3 23 38 2B D9 33 F0 8E 7F 0E 25 6B C8 24 9E 40 95 B1 EC**
PCR_6	**B2 A8 3B 0E BF 2F 83 74 29 9A 5B 2B DF C3 1E A9 55 AD 72 36**
PCR_7	**AB 3D 75 4C 40 CF C0 78 99 8F A3 E1 A6 92 C6 01 67 92 F1 C5**

.
.
.

Up To **PCR_{23}** Now,

CoT_{d0} = { PCR_0

= 663F8EE75A5C33FDEE81AF20FA9E7F651F479135,
(298df125b260ef64201bdf0815c003873eedd50e [BIOS:EV_S_CRTM_VERSION],
B794d4c2eddf03b69bdc8ac0a1cbf9278fe131ca0 [BIOS:EV_POST_CODE(EV_CODE_NOCERT)]
.
.
(name and value pairs for PCR_0))

PCR_1 = BBC045C19F21F54F49B13895EC49A9ECBF15F81D,
PCR_2 = 0CE8061EBA5C532C62595DDB3075CAE8A8575361, … Up To PCR_{23} }

Similarly , we received following CoT for domain-U

CoT_{dU} = {PCR_0 = 89A05EBEF181D53653680BDB59C75DC1E06AB2AB,
PCR_1 = AB8BA78A38E836D77AAA595F2A27A57E87090928, PCR_2 =
D32F78E10FC41274DC447AE01576FD547F2E4E36, …, Up To PCR_{23}}

Figure 8: PCR values and its corresponding domain CoTs

References

[1] A4Cloud, *The Cloud Accountability Project*, July 3, 2016. (http://www.a4cloud.eu/)

[2] I. M. Abbadi, "Clouds trust anchors," in *IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'12)*, pp. 127–136, 2012.

[3] M. Almorsy, J. Grundy, and I. Müller, "An analysis of the cloud computing security problem," in *Proceedings of APSEC Cloud Workshop*, pp. 8–18, 2010.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.

[6] D. Catteddu, M. Felici, G. Hogben, A. Holcroft, et al., "Towards a model of accountability for cloud computing services," in *International Workshop on Trustworthiness, Accountability and Forensics in the Cloud (TAFC'13)*, pp. 1–10, 2013.

[7] D. Contractor and D. Patel, "Analyzing trustworthiness of virtual machines in data-intensive cloud computing," in *2014 Twelfth Annual International Conference on Privacy, Security and Trust (PST'04)*, pp. 403–406, July 2014.

[8] J. G. Dyer, M. Lindemann, R. Perez, R. Sailer, L. Van Doorn, and S. W. Smith, "Building the IBM 4758 secure coprocessor," *Computer Journal*, vol. 34, no. 10, pp. 57–66, 2001.

[9] IMA, *Integrity Measurement Architecture (Linux IMA)*, July 3, 2016. (http://sourceforge.net/apps/mediawiki/linux-ima/index.php)

[10] T. Mather, S. Kumaraswamy, and S. Latif, *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. O'Reilly Media, 2009.

[11] F. Mayer, D. Caplan, and K. MacMillan, *SELinux by Example: Using Security Enhanced Linux*. Pearson Education, 2006.

[12] P. Mell and T. Grance, "The NIST definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, no. 6, pp. 50, 2009.

[13] A. Nakhimovsky, T. Myers, *Google, Amazon, and Beyond: Creating and Consuming Web Services*, Springer, 2004.

[14] D. Nunez, C. Fernandez-Gago, S. Pearson, and M. Felici, "A metamodel for measuring accountability attributes in the cloud," in *IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, vol. 1, pp. 355–362, 2013.

[15] S. Pearson, "Towards accountability in the cloud," *IEEE Internet Computing*, vol. 15, no. 4, pp. 64–69, 2011.

[16] S. Pearson, V. Tountopoulos, D. Catteddu, et al., "Accountability for cloud and other future Internet services," in *IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom'12)*, pp. 629–632, 2012.

[17] R. Perez, R. Sailer, and L. van Doorn, "vTPM: Virtualizing the trusted platform module," in *Proceedings of the 15th Conference on USENIX Security Symposium*, pp. 305–320, 2006.

[18] V. Scarlata, C. Rozas, M. Wiseman, D. Grawrock, and C. Vishik, "TPM virtualization: Building a general framework," in *Trusted Computing*, pp. 43–56, 2008.

[19] J. Schiffman, T. Moyer, H. Vijayakumar, T. Jaeger, and P. McDaniel, "Seeding clouds with trust anchors," in *Proceedings of the 2010 ACM workshop on Cloud Computing Security Workshop*, pp. 43–46, 2010.

[20] J. Singh, "Cyber-attacks in cloud computing: A case study," *International Journal of Electronics and Information Engineering*, vol. 1, no. 2, pp. 78–87, 2014.

[21] S. B. Sitkin and N. L. Roth, "Explaining the limited effectiveness of legalistic remedies for trust/distrust," *Organization Science Journal*, vol. 4, no. 3, pp. 367–392, 1993.

[22] S. W. Smith, *Trusted Computing Platforms: Design and Applications*, vol. 2, Springer, 2005.

[23] I. Srivastava, *South Korea Cyber-attacked on Korean War Anniversary*, June 29, 2013. (http://timesofindia.indiatimes.com/world/rest-of-world/South-Korea-cyber-attacked-on-Korean-war-anniversary/articleshow/20833246.cms?referral=PM)

[24] Trend Micro, *Trend Micro Investigates June 25 Cyber Attacks in South Korea*, July 1, 2013. (http://www.trendmicro.com/vinfo/us/threat-encyclopedia/web-attack/124/trend-micro-investigates-june-25-cyber\-attacks-in-south-korea)

[25] Trusted Computing Group, *TCG Infrastructure Working Group Reference Manifest (RM) Schema Specification, Version 1.0*, Nov. 16, 2006. (https://www.trustedcomputinggroup.org/wp-content/uploads/IWG-Reference_Manifest_Schema_Specification_v1.pdf)

[26] Trusted Computing Group, *TCG Software Stack (TSS) Specification, Version 1.2*, July 3, 2016. (http://www.trustedcomputinggroup.org/tcg-software-stack-tss-specification/)

[27] Z. Wang, Y. Lu, G. Sun, "A policy-based deduplication mechanism for securing cloud storage," *International Journal of Electronics and Information Engineering*, vol. 2, no. 2, pp. 70–79, 2015.

[28] J. Yao, S. Chen, C. Wang, D. Levy, and J. Zic, "Accountability as a service for the cloud," in *2010 IEEE International Conference on Services Computing (SCC'10)*, pp. 81–88, 2010.

[29] A. R. Yumerefendi and J. S. Chase, "Trust but verify: accountability for network services," in *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop*, pp. 37, 2004.

[30] M. Zareapoor, P. Shamsolmoali, and M. A. Alam, "Establishing safe cloud: Ensuring data security and performance evaluation," *International Journal of Electronics and Information Engineering*, vol. 1, no. 2, pp. 88–99, 2014.

[31] L. J. Zhang and Q. Zhou, "CCOA: Cloud computing open architecture," in *IEEE International Conference on Web Services (ICWS'09)*, pp. 607–616, 2009.

Dipen Contractor Dipen contractor received his B.E degree in computer engineering in 2008 & M.E. degree in computer science and engineering in 2011. He is pursuing Ph.D. in computer engineering at National Institute of Technology Surat. His research interests include Cloud computing, Information security, Trust management, Remote attestation, and programming with Trusted Platform Module.

Dhiren Patel Dr.Dhiren Patel is a professor of computer engineering at National Institute of Technology Surat. He carries more than 20 years of experience in Academics, Research & Development of Secure ICT Infrastructure Design. His research interests cover Security and Encryption Systems, Cloud Computing and IoT, Identity and Access Management, e-Voting, Advanced Computer Architecture etc. Besides numerous journal and conference articles, Prof. Dhiren has authored a book "Information Security: Theory & Practice" published by Prentice Hall of India (PHI) in 2008. He is actively involved in Indo-UK, Indo-Norway, and Indo-Japan security research collaborations.