Towards Corporate Confidentiality Preserving Auditing Mechanisms for Clouds

Roland Bless, Matthias Flittner
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: {bless, flittner}@kit.edu

***Abstract*—IT services realized within cloud environments often suffer from a lack of transparency. On the one hand this opacity with respect to operational practices protects business secrets of a cloud provider. On the other hand, if services from critical infrastructures run within clouds this opacity becomes a severe problem, e.g., for a later root cause analysis in case of failures. This paper presents approaches for auditing the mapping of virtual resources to physical resources while preserving corporate confidentiality of the cloud providers. We also make sure that the resulting auditing information is useful for a later root cause analysis. Our research concludes that statistical binding and computational hiding commitment schemes are a suitable solution. Moreover, our approach enables a trusted third party to perform audits on behalf of tenants.**

***Keywords*—Cloud Security, Auditing, Non-Repudiation, Commitments, Transparency**

I. Introduction

Cloud Computing is not only a major recent trend in the IT industry, it has also started to revolutionize the way enterprises create and deliver their IT solutions. As more sectors use cloud-based services in their computing environments, the *Critical Infrastructure (CI)* sector will likely also adopt this approach for some of its services. This trend is confirmed by the fact that telecommunication providers are currently considering to move telecommunications functions into the cloud, e.g., the ETSI ISG NFV (Industry Specification Group – Network Functions Virtualization) was formed in 2013 with the purpose of developing pre-standards. Anticipated advantages for running CI services in the cloud range from cost reductions and increased flexibility to new ways for improving the resilience and availability of the CI, e.g., through the use of abundant virtual resources. Running IT services for CI in the (public/community/hybrid) cloud implies several security and resilience requirements that existing cloud offerings do not address well. Due to the opacity of cloud environments (e.g., actual location of virtual machines or stored data), the risks of deploying cloud-based CI services are difficult to assess, especially at the technical level, but also from legal or business perspectives. Existing security measures do not address related important issues (e.g., risk, trust, and resilience), resulting in uncertainty for operators and manufacturers of critical infrastructure IT systems. Moreover, the general need for extra measures to increase trust and accountability in clouds is also stipulated by others [1], [2], [3].

Additionally, the CI sector characterized strict regulatory constraints often originating from legal requirements for liability. This results, for example, in a need for technical means to support evidence and data protection laws. In case of a failure of CI services, a *Root Cause Analysis (RCA)* should provide evidence who is liable for this failure. In some cases legal requirements need that CI service providers – now being cloud users – must assure the compliance with specific policies even if their services are running inside a cloud. From a legal point of view, trusting the *Cloud Infrastructure Provider (CIP)* to adhere to negotiated requirements is not sufficient in many cases. Thus, the CI service provider must have the possibility to assure himself by independently gathering evidence or performing audits for the cloud infrastructure. This demand is in contrast to the CIP's objectives of keeping operational practices and details about the cloud infrastructure confidential as well as to have the freedom and flexibility in managing cloud resources.

In this paper we present mechanisms that allow to verify the cloud infrastructure provider's adherence to some agreed policies (e.g., virtual resource anti-affinity) that play an important role for CI services without sacrificing a cloud infrastructure provider's corporate confidentiality.

A. SECCRIT architectural framework

Before describing the approach in detail it is necessary to clarify the roles, concerns, and responsibilities of the involved stakeholders by using an architectural framework developed within the SECCRIT project [4], [5]. An analysis of existing architectural frameworks from the viewpoint of a CI service provider showed that most of them assume a rather monolithic role of the "Cloud Service Provider", whereas it is identified the need for applying a more precise role distinction of stakeholders in this context, motivated by liability questions. Many existing architectural models do not offer a clear separation of administrative concerns, lack of support for additional monitoring/metering capabilities and interfaces required for running CI services in cloud environments.

Consequently, Fig. 1 shows different levels of abstraction that we need to distinguish. The different abstraction levels correspond to different stakeholders and their view on the managed resources. Most existing architectures concentrate on the provisioning of resources at the Cloud Infrastructure level, i.e., how to provide and manage cloud resources within a data center. A more overarching view covering different levels and stakeholders is needed: At the top level, i.e. *User Level*, we have the CI Service User who remotely accesses the CI Service. The next lower level is controlled by the *CI Service Provider* who manages the resources at the Service Level. The CI Service is composed of several components

978-1-4799-2730-2/14/$31.00 ©2014 IEEE

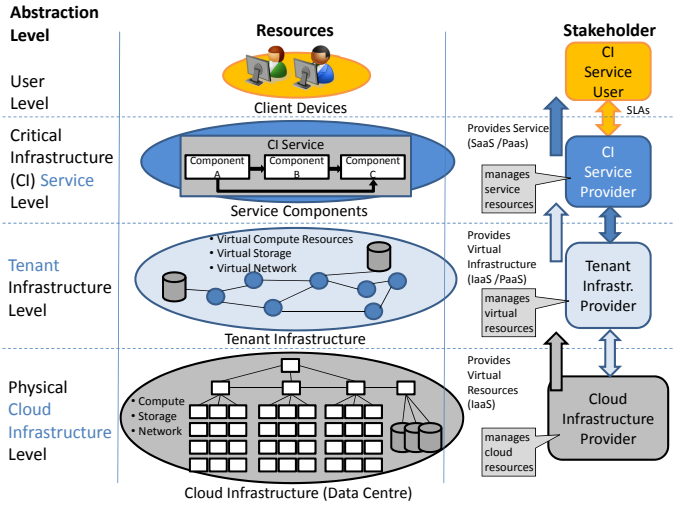

Fig. 1. Different considered abstraction levels within the SECCRIT architectural framework

that interact with each other in order to provide the actual service. The service components are instantiated on the virtual infrastructure that is provided by the next lower level. The *Tenant Infrastructure Level* provides a virtual infrastructure that consists of virtual compute, virtual storage, and virtual network resources. We distinguish this stakeholder from the service provider since they may be separate organizations. Several such tenants are typically hosted within one Cloud Infrastructure, which is the next lower level. The *Tenant Infrastructure Provider (TIP)* is not aware of what services and applications are running inside his virtual infrastructure. The *Physical Cloud Infrastructure* Level at the bottom level provides real physical compute, storage, and network resources, which are hosted in a data center and administered by the CIP. This level usually provides virtual resources to its upper level, i.e., the tenant. The virtualization solution usually provides (a certain degree of) isolation between the different tenants that are multiplexed onto the same physical infrastructure and thus permits the sharing of resources. For increasing the resource efficiency or performing maintenance tasks, the CIP can usually transparently move virtual resources across its physical infrastructure, unnoticed by the tenant. However, for a tenant it would be helpful to obtain evidence about such migrations in case that they entail a service failure or a violation of a non-functional constraint.

It should be noted that multiple stakeholder roles may sometimes be realized by the same organization. Moreover, we need several different views within the framework suitable to focus on specific aspects that need to be investigated. Those additional views are explained in a recently published white-paper [5], which also contains a more detailed discussion of legal aspects, a gap analysis, and also examples of how to map existing cloud virtualization platforms into the architectural framework. In SECCRIT we consider the CIP as being trustworthy in the sense that he will not attack his own customers, i.e., tenants. Since the CIP has unfettered access to all physical and virtual resources, he has absolute control over all data and resources of all tenants. While other projects like TClouds [6] considered how to increase trust to cloud platforms, we do not assume that the CIP performs malicious actions against his customers. In the long run, this would put him out of business if such doings are revealed. Nonetheless, a disgruntled employee of the CIP might act maliciously for his personal benefits, e.g., stealing data or attempting to influence service behavior; the internal authorization and monitoring infrastructure of the CIP must prevent or log such insider attacks. Cloud Auditors [7] may help to check for such actions, and discover and reveal any dubious provider practice.

B. Objectives and Challenges

The previous section motivated that it is necessary to develop techniques which are helping TIPs to get a *more detailed view* on their virtual environment that is run by the CIP. Such a detailed view is essential for auditing purposes that the tenant is interested or legally obliged in. A further requirement is that the auditing information must contain enough *evidence* to be used in court or for an RCA. Therefore, the obtained information from audits should not be changeable in retrospect. Finally, the information provided by a CIP to TIP should of course *not disclose* any confidential business information such as information about other tenants or about the CIP's operational practices. We want to keep open the possibility to transfer the auditing process to a *Trusted Third Party (TTP)* which could act on behalf of a tenant. In this case the tenant is relieved from supervising the auditing process.

In this paper we are focusing on scenarios in which the TIP is particularly interested in affinity or anti-affinity constraints (e.g., physical host affinity, firewall rules, security posture). For example, we assume that the TIP wants to check if two virtual resources are not placed onto the same physical cloud node. Anti-affinity could be useful, besides legal requirements, if two virtual resources should be used in a high-availability scenario. We consider that this requirement must be auditable in a way such that the resulting information is usable for a later RCA, is non-repudiable while keeping corporate confidentiality. Additionally, it should be possible to transfer this procedure to a TTP as auditor.

II. Related Work

There exists a lot of work related to Cloud Computing in similar areas like trust, auditing, monitoring, and RCA. Nevertheless, no current solution enhances transparency of virtual resource location and protects private information of a CIP at the same time.

A. Trust Framework for Cloud Computing

Ko et al. [8] describe a framework to raise trust and accountability in cloud scenarios for tenants. Their framework is divided into abstraction levels which are called workflow layer, data layer, system layer, policies and law & regulations. They propose to use detective approaches that help tenants to detect *Service Level Agreement (SLA)* violations. These approaches are less invasive as preventive solutions, since only detection mechanisms are needed and no interception has to be done. Furthermore, the authors argue that post-mortem analysis is as important as real-time approaches are. Overall, they focus on post-mortem analysis of log information and stored pieces and not on run-time checking of SLA guarantees.

Therefore, their paper shows a way to enhance trust for could computing by enabling a post-mortem analysis of stored information that is an essential precondition to run an RCA. For this reason, we agree that it is very important to store information in a proper way so that a post-mortem analysis is meaningful. Nevertheless, the approach is not focusing on real-time auditing of requirements like affinity or anti-affinity of virtual resources. Moreover, our approach focuses on auditing requirements and allows for using the results in a later RCA.

B. CloudTrust Protocol

Another work which is relevant for our publication is the CloudTrust Protocol (CTP) from the Cloud Security Alliance [9], first described in [10]. Focus of this approach is to develop a protocol that enhances trust in cloud scenarios. They claim that digital trust can only be achieved by eliminating the lack of transparency in cloud computing. Therefore, they introduce a protocol that enables tenants to ask CIPs about the configuration, vulnerability, access, authorization, policy, accountability, anchoring, and operating status condition of their virtual resources. The CTP includes elements of transparency which are identity/session, configuration, vulnerability, anchoring, audit log, service management, users & permissions, configurations, and quotas. CTP describe how to ask about an element of transparency, how to package the answer, and defines the data format and interfaces.

However, the CTP itself describes no implementation of functions that should be supported. Moreover, it is assumed that to check affinity or anti-affinity the concrete mapping of virtual resources has to be disclosed. The only proposed alternative is that the CIP itself checks the requirements and returns if they are fulfilled or not. The CTP approach does not consider whether the return values are authentic enough for an RCA or a debate in court. Therefore, we introduce in our paper an implementation possibility to allow checking the affinity or anti-affinity conditions while keeping the possibility to run a post-mortem RCA.

C. Cloud Audits

An approach of Bryan Ford et al. [11] tries to audit *cloud reliability* while preserving private information of the CIP. This is very useful because replication itself does not guarantee reliability. Virtual resources must additionally be structural redundant in order to guarantee reliability so that underlying hardware is independent from each other. The core idea of their paper is to generate dependency fault trees of infrastructural components such as power supplies. Such fault trees are customized with a secure multi-party computation technique [12] so that no private information of the CIP is included in these trees. But these fault trees contain enough information so that the tenant can identify weak points of structural dependencies. In this manner tenants could check the reliability guarantees of their multi-provider cloud setup without gaining sensitive information of the CIP. However, this solution only applies to fault trees of structural dependencies and common multi-provider setup but is not applicable to location requirements of virtual resources. Nevertheless, the idea to protect business secrets of the CIP while simultaneously enabling tenants to verify the fulfillment of their requirements is similar to our main objective.

Other solutions try to verify that the CIP stores the mission data correctly and tracing and logging all significant events. Kan Yang et al. gives a good overview of existing techniques and challenges [13]. Unresolved problems in this area are how to audit dynamic data and how to design the auditing process efficiently. Several approaches consider a third party auditor to perform audits on behalf of tenants. Data privacy is of particular importance in this context. Cong Wang et al. [14] address this problem and consider communication and computational overhead in their approach. During the whole auditing process the TPA would not get any knowledge about the data content stored in the cloud. Instead of using encryption and transferring the data to the third party auditor they are using a homomorphic linear authenticator [15] which enables the TPA to perform auditing without demanding a local copy of the data and without any knowledge of the data. The auditor only checks if the confidential data is stored properly. The focus is on solutions that verify the fulfillment of tenants' requirements in a cloud environment without disclosing business secrets of the CIP. These techniques are a great way to enhance transparency of cloud computing.

Nevertheless, all solutions introduce the idea of non-invasive auditing cloud environments and aim at ensuring corporate confidentiality. However, these techniques cannot be used to audit affinity or anti-affinity without disclosing unwanted information. Additionally, the focus on non-repudiation and later RCA is missing in those papers. Our paper is focusing on these aspects and tries to achieve both: real-time auditing of requirements and the possibility to use the results for RCA or in a court, which includes non-repudiation.

III. Mapping between Virtual Resource and Physical Nodes

In order to make the location of virtual resources somewhat transparent for tenants while preserving corporate confidentiality (about operational practices available resources and so on), a mapping between virtual resources and physical nodes is needed. A tenant gets identifiers v_i for his virtual resources from the CIP in order to control and manage them in the tenant infrastructure management system. The tenant infrastructure operator would like to verify that the mapping of a virtual resource v_i to a physical resource p, is conforming to the constraints. For instance, if p, q are unique identifiers for physical resources, then a tenant may want to check whether the mapping for two virtual machines v_i, v_j holds an *anti-affinity condition*, such as if (v_i, p) then (v_j, q) with $v_i \neq v_j \wedge p \neq q$.

Moreover, neither the CIP wants to expose details about the used physical infrastructure nor the tenant is interested in the particular mapping, but rather in verification of the specified constraints. On the one hand, the CIP usually wants to be free in his decision about when, how, and where to migrate virtual resources between physical hosts. On the other hand, the tenant does not care about migration actions as long as they do not lead to service failures or violations of availability constraints. Thus, for instance in case of a service failure at point in time t, a tenant in possession of a mapping at time t has got an *evidence* that a root cause for the failure is related to a migration action of the CIP that violated given constraints.

Some unwanted private information of the CIP could leak if physical identifiers are exposed to the tenant. For instance, ten-

ants could be able to identify concrete physical hosts with this identifier and could observe migration behavior or calculate the amount of used or available physical hosts. Similarly unwanted is that if tenants could correlate this identifier with other tenants checking whether their virtual resources run on the same hardware as their competitor's virtual resources do. This information is very useful for launching attacks against these competitor resources, e.g., in so-called side-channel attacks [16]. Though we assume that the CIP is trustworthy and not acting maliciously against its tenants, we want to make sure that the evidence that a tenant gathers is also trustworthy and credible. Therefore, we conclude that a solution is needed that allows a tenant to check conditions about resource mappings under the following constraints:

- *on-demand and run-time checking* of mapping conditions — the tenant should have the possibility to check the fulfillment of agreed mapping conditions at any time.
- *authenticity and non-repudiation* of the mapping — the mapping should be authentic in the sense that the CIP cannot deny the existence of a particular mapping (v, p) at point in time t or change it afterwards, e.g., stating that it actually was (v, q) at t.
- *avoid unnecessary disclosure* of physical resource topology and characteristics, about other tenant resources, operational practices and so on.
- *possibility to transfer auditing* process to a TTP so that the tenant itself is not involved in auditing or storing information.

Consequently, the CIP needs to provide management interface functions that allow the exchange of such information on demand. Moreover, mappings like (v, p) between virtual resource v and physical node p must be authenticated and protected against later manipulation. Additionally, the physical identifier must be designed in such a way that no private information about the CIP and other tenants can be derived.

The non-repudiation requirement could be solved with current state of the art solutions for authentication like signing and TTP timestamping with the help of a *Time Stamping Authority (TSA)* [17]. For this paper we assume that an authenticated mapping for a virtual resource consists of a CIP signed tuple like $(v, p, t_{\mathrm{CIP}})_{\mathrm{sigCIP}}$ where $\mathrm{t_{CIP}}$ stands for a CIP timestamp of this mapping. Also the mapping contains a TSA timestamp t_{TSA} together with a signature $(\mathrm{sigCIP}, t_{\mathrm{TSA}})_{\mathrm{sigTSA}}$ that binds the CIP signature for the tuple (v, p, t_{CIP}) with the TSA timestamp t_{TSA}. That means the CIP signs the existence of a mapping (v, p) at point in time t_{CIP}. This signature sigCIP is sent to the TSA that signs it together with its own timestamp t_{TSA}. The offset between both time bases should not be too large, i.e., the difference between t_{CIP} and t_{TSA} should be small in order to stay credible.

These authenticated mappings between virtual identifier, physical identifier and timestamp will be provided by the CIP on request of a tenant. Given $<(v, p, t_{\mathrm{CIP}})_{\mathrm{sigCIP}}, (\mathrm{sigCIP}, t_{\mathrm{TSA}})_{\mathrm{sigTSA}}>$ the tenant or any other third party can verify the binding between the mapping (v, p) and the point in time t_{CIP}. Finally, the CIP cannot repudiate the existence of the mapping (v, p) at point in time t_{CIP} and the tenant cannot claim to have seen a different mapping, e.g., (v, q) instead of (v, p). Moreover, we assume that the CIP will provide the mapping information correctly. Scenarios in which the CIP is acting maliciously against its own tenants by forging mappings during run-time are not considered in this paper. It is sufficient to assume that a CIP that works according to current standards and is under ongoing certification implies a sufficient level of trust. For stronger assumptions trusted platform modules and other techniques may be needed.

IV. Design of Non-Disclosing and Non-Repudiable Physical Identifiers

The physical identifier p of the tuple (v, p, t_{CIP}) should be constructed in such a way that a tenant could compare it with other own physical identifiers to audit affinity or anti-affinity while preserving corporate confidentiality at the same time. It is obvious that the physical ID cannot be a CIP globally unique ID, because otherwise a tenant that came into possession of a competitor's physical ID could compare it with it's own resources identifiers. In this case the tenant could use or try to deploy own virtual resources on the same physical host in order to attack the physical host resources or the virtual resources of the competitor. Moreover, a tenant could reconstruct the migration strategy and the number of physical hosts a CIP uses, because the ID would change for every migration. Both the operational practices of the CIP and the total amount of used physical hosts are business secrets that should not be disclosed to tenants. Additionally, the physical ID must allow to identify a certain physical host post-mortem for an RCA or court debate.

A. Hashed Physical Identifiers

A common possibility is to generate pseudo physical identifiers by using a hash function $H(\cdot)$. This hash function should be unique per tenant, since otherwise tenants can again correlate their physical identifiers with others and use them for attacks. For this reason the hash function should be enriched with a tenant specific identifier c. OpenStack actually follows this approach, i.e., the keyed hash function for physical IDs looks like $H(p, c)$. Thus, instead of returning the actual physical ID p, the CIP returns the result of the hash function $h := H(p, c)$ if asked by the tenant for the current mapping of virtual resource v, resulting in a mapping tuple (v, h, t_{CIP}) from a tenant point of view. If, for example the virtual resource v_x from tenant c_a is placed on the physical node p_y the mapping tuple should look like (v_x, h_1, t_1) in which h_1 stands for $H(p_y, c_a)$ and t_1 for the CIP timestamp. In addition, the used hash function should satisfy some properties like a cryptographic hash function does. These properties are: efficiently to compute, deterministic, pre-image resistance, second pre-image resistance and collision resistance [18].

Furthermore, the CIP has to save input values of the hash function in such a way that an RCA could be performed later. Similarly, the TIP has to save the mappings to supply the information if it is needed for an RCA or in court. However, per tenant hashes do not solve the problem satisfactorily. Although the physical ID is hashed so that it can no longer be compared with other tenants, it still contains information about

the total number of physical hosts and migration strategy of the CIP.

A remedy could be to use a kind of a nonce, which is only valid for a short time period and to put this additional information into the calculation of the hash value. We call this *epoch nonce* e that should be valid for a short time only (the epoch T_e), e.g., for a few minutes. During this period this nonce is constantly hashed with the tenant and the actual physical node identifier p like $H(p, c, e)$. The nonce e should be randomly chosen, must be stored by the CIP, and disclosed to the tenant. If the tenant wants to check affinity of virtual resources v_1 and v_2, their pseudo physical IDs must be requested within a short period of time, ideally within the same epoch T_e. Thus, it is achieved that the CIP uses the same epoch nonce to generate the pseudo physical IDs. In this case $H(p_1, c_a, e_x)$ and $H(p_2, c_a, e_x)$ are comparable, because the epoch nonce e_x is the same. If p_1 and p_2 are identical then both hash values are same. But if the epoch nonce changed, then hash values $H(p_3, c_a, e_x)$ and $H(p_4, c_a, e_y)$ are incomparable. Consequently, if a tenant asks for the pseudo physical IDs of virtual resources v_1 and v_2 the CIP should provide not only pseudo IDs, but also the used epoch nonces.

The problem with this system is that the epoch T_e must be large enough that a tenant could sent his request but small enough that he could not request the whole physical identifiers of all virtual resources. An obvious modification solves this problem. Instead of using epoch nonces the CIP could use random request nonces n. Now, the customer must request all candidates for which the physical IDs are needed at once. For each request the CIP could use a new random request nonce and put it in the calculation of the hash values. For example a tenant could request the physical IDs of two virtual resources, the CIP chooses a random nonce n and calculates the identifiers $H(p_1, c_a, n)$ and $H(p_2, c_a, n)$. For tenants it is impossible to compare identifiers from different requests. Thus, migrations are now undetectable, but a tenant is still in a position to count per request all underlying physical nodes, because per request the same random nonce is injected into the hash functions. Moreover, the CIP has to store all information that was used to calculate the hash values so that he could prove authenticity of the hash values in a post-mortem analysis.

Finally, a cryptographic hash function never fully protects against extraction of any useful information from the hash values about the original message. Imagine a hash function that is keeping specific bits of a message in the resulting hash value. There is no guarantee for perfectly hiding the hashed information. Only typical one-way function guarantees like computational difficulty of complete inversion can be achieved. In other words, it is possible that parts of the hashed information are leaking from the hash value even if the hash function is a strong cryptographic hash function. Therefore, a hash function should deterministically and efficiently computable and behave as much as possible like a random function [19]. A current state-of-the-art candidate for such a hash function is the SHA-3 winner called Keccak [20] that is currently assessed as being secure enough to protect all digits of a hashed value.

B. Committed Physical Identifiers

Commitments must satisfy two properties: *hiding* and *binding* of the committed value. Whereby one property could be achieved with perfectly statistical strength and the other only with computational strength. However, one can choose which requirement is to fulfill with which strength. The difference to hash functions is that *hiding* means here that it is difficult to gather any useful information from values, whereas *binding* means that the archetype of a commitment is unique. Another great advantage is that per default *blobs* (i.e., committed values) are enriched with *fresh randomness*. Basically, a commitment scheme could be constructed from a one-way function, hash function, or encryption scheme.

1) Perfect Hiding – Computational Binding: A commitment scheme that offers perfect hiding and computational binding of committed values are Pedersen commitments [21]. Their binding property is based upon the discrete logarithm assumption. For this scheme a cyclic group G and two generators g and h are needed. The assumption holds that the discrete logarithm of h to the base g is computational difficult, which means practically unfeasible. If the CIP wants to commit a mapping of a virtual resource v to a specific physical node ID p he has to choose a random value r and calculates the *blob* $b := \mathrm{commit}(p, r) := g^p \cdot h^r$. If, once necessary, the committed value could be disclosed by providing p and r for example $\mathrm{unveil}(b) := p, r$. The security of this scheme depends on the hiding and binding properties. In this example no adversary (i.e. tenant) could compute the physical identifier from a blob. The CIP could not forge another value into the blob without solving the difficulty of the discrete logarithm problem.

Mappings between virtual resources and physical identifiers are now constructed with the help of commitments. In a first step the CIP has to generate blobs for physical node IDs where the virtual resources reside on. If for example virtual resource v is on physical node p the commitment to this value should look like $b := \mathrm{commit}(p, r) := g^p \cdot h^r$ whereby r is a random nonce. This blob is now substituted into the mapping tuple like (v, b, t_{CIP}). At a later time, the tenant can request unveiling the blob through the CIP. The CIP has then to provide p and r to prove the original content of the commitment. Only with negligible probability he is able to provide wrong values that match the blob.

It is possible to use this commitment scheme to solve the problem with the physical IDs like keeping corporate confidentiality while still having the opportunity to check affinity or anti-affinity of these identifiers. The private information of the CIP is well protected, because a tenant could not get any useful information out of a blob (committed value) as long as it is not disclosed; this is guaranteed through the perfect hiding property. The disadvantage is that the values committed into blobs are no longer directly comparable, because this could be a useful information that is protected by the hiding property. Therefore, a mechanism is needed that can assess anti- or affinity of committed values in blobs. The hiding property has to be given up a little bit.

With Pedersen commitments it is simple to show *affinity* using the *masking property* of this scheme. Masking of Pedersen means that there is a easy way to compute a new commitment on the same value as previous one. The first commitment on a value would be calculated as described above $b_1 := \mathrm{commit}(p, r_1) := g^p \cdot h^{r_1}$. If a second commitment should be created on the same value, the original commitment is enriched with fresh randomness r_2 like $b_2 := b_1 \cdot h^{r_2} =$

$g^p \cdot h^{r_1} \cdot h^{r_2} = g^p \cdot h^{r_1+r_2}$. Therefore, b_1 and b_2 are blobs on the same committed value p_1. If the CIP wants to prove that to the TIP, he has to disclose r_2. The TIP itself could then check if $b_2 \stackrel{?}{=} b_1 \cdot h^{r_2}$. If this condition holds, both blobs contain the same value p_1. The TIP learns nothing about the content of the commitments except affinity if they are affine and if and only if the CIP provides the correct r_2.

The second requirement, *anti-affinity* of two blobs is a bit more difficult to prove. Therefore, it is only conceptually shown in this paper. But the requirement is basically provable with a set of *non-interactive zero-knowledge (NIZK)* proofs. In principle one has to show that for the division of two commitments $D := \frac{b_1}{b_2} = g^{p_1-p_2} \cdot h^{r_1-r_2}$ an i exists so that the result of the division to the power of i is of the form $D^i = (g^{p_1-p_2} \cdot h^{r_1-r_2})^i = g^1 \cdot h^{(r_1-r_2)\cdot i}$. It is sufficient to prove the existence of an i (being an inverse multiplicative element for $p_1 - p_2$) through an NIZK proof of knowledge that keeps the i secret. It is obvious that such an i could not exist if p_1 and p_2 are identical. In such a case the difference is 0 and thus $(g^{p_1-p_2} \cdot h^{r_1-r_2})^i = 1 \cdot h^{(r_1-r_2)\cdot i} \neq g^1 \cdot h^{(r_1-r_2)\cdot i}$ holds. The technique of Abe et. al [22] is one possibility to proof the multiplication of commited values. Therefore, the CIP has to proof that he is able to open a commitment $D^k \cdot h^l$ that is using the difference D as g and that the same commitment opens to the value $g^1 \cdot h^{r_*}$ if using g instead of D. Therefore, only if the physical IDs are different, a proper proof could be provided by the CIP. A thorough mathematical sound computational complexity investigation is left as future work.

Nevertheless, by using commitments it is possible to audit (anti-)affinity of virtual resources while completely protecting company secrets. Additionally, the blobs contain enough information for a post-mortem RCA or a debate in court. Solely combinatorial information about the CIP's resource setup could leak like number of simultaneously used physical hosts. But it is impossible to learn migration strategies, to gather information about other operational practices of the CIP, or to count the total amount of used physical hosts. The amount of information CIP and TIP have to save should be reduced. Usually, the CIP has to save all inputs and outputs of the commitment functions and the TIP has to save the mappings. Unfortunately, if a TTP should perform auditing, all this information has to be transferred to this party. Finally, it is worse if the binding property has only computational strength, because a CIP has a lot of computing resources and may be able to try forging a commitment. The hiding property is, however, perfectly statistical and a tenant has less resources available, too.

2) Computational Hiding – Perfect Binding: As mentioned before, it would be much better if the properties were the other way round so that the commitment scheme is computational hiding and perfect binding. Such a scheme is unforgeable for the CIP and securely hiding if the TIP has no huge computational power. A further advantage is that it is possible to put a trapdoor into the scheme so that a TTP with knowledge of the trapdoor could break the hiding property. Besides that by using the trapdoor it is no longer necessary to save all inputs to unveil the commitments, since knowledge of the trapdoor is sufficient to solve this. However, one must be careful that the tenant does not get hold of the trapdoor information. One commitment scheme which offers all these properties is the *ElGamal commitment scheme*, which is a modification of the ElGamal encryption scheme [23]. It is very similar to the above-introduced scheme from Pedersen, but with reverse characteristics and a built-in trapdoor to break the hiding property. The hiding property is based on assumptions about the difficulty of discrete logarithms and on the Diffie-Hellman assumption as ElGamal encryption scheme is [24]. It is obvious to choose a strong variant. The binding property therefore is perfectly statistical. The ElGamal commitment scheme is based on a cyclic group G and a generator g. A second generator h is created with a random secret x like $h := g^x$. In the case that a TTP should be able to open commitments, the secret x has to be transferred to this party. It would be even better if the TTP sets the scheme up and transfers only G, g, and h to the CIP while keeping the secret x. The CIP can therefore commit to physical IDs very similarly as in the previously described Pedersen commitment scheme.

First, he has to choose a random nonce r. Then he calculates a part of the blob $z_2 := \text{commit}(p, r) := g^p \cdot h^r = g^p \cdot (g^x)^r = g^p \cdot g^{x \cdot r}$, whereby p is the physical identifier and r a random nonce. Additionally, he has to calculate the second part of the commitment called z_1 that contains special information for using the trapdoor $z_1 := g^r$. The trapdoor is very similar to a Diffie-Hellman key exchange on which the ElGamal encryption scheme is based [25]. Both parts z_1 and z_2 form the blob $b := (z_1, z_2)$. This blob b for a physical identifier p of a virtual resource v is placed into the mapping as described above in the Pedersen example like (v, b, t_{CIP}). This mapping will be transferred to the requesting tenant. If now the need arises to reveal the commitment in retrospect, the CIP could provide the stored p and r to prove the content of the blob, while only using the second part of the blob z_2. Verification is done in the same way as described for Pedersen commitments. However, the commitment can be unveiled via a second way using the trapdoor, which at best is only known to a TTP. To open the blob with the trapdoor x one has to take the first part $z_1 := g^r$ of the commitment and to calculate $s := (z_1)^x = (g^r)^x = g^{r \cdot x}$. After that it is possible to calculate the content of the commitment from the second part like $\frac{z_2}{s} = \frac{g^p \cdot g^{x \cdot r}}{g^{r \cdot x}} = g^p$. Consequently, the committed value p is not directly returned, but only g^p. To solve this problem the CIP has to provide a mapping table from physical identifiers p to g^p. This additional transfer mapping could be also stored on a TTP like the trapdoor x. In this manner no additional information except the secret key x and the transfer mapping has to be stored to open a commitment or to prove containing content. Note that now the binding property is perfectly statistical, because with knowledge of the secret x the provided values p and r could be verified. The CIP could not forge later another physical identifier into a commitment.

Affinity of two committed physical identifiers could be shown as described for Pedersen commitments by using the masking property. So if (z_1, z_2) and (z_3, z_4) should be shown as being identical commitments on same physical identifiers, the CIP has to provide a r_* which is the transition from z_2 to z_4 like $z_4 = z_2 \cdot g^{r_*}$. Note that he used this r_* originally to calculate the commitments. Anti-affinity of the content of two blobs should be shown in the same way as for Pedersen commitments by providing a set of NIZK proof that an i exists that will transform $(\frac{z_2}{z_4})^i$ to $g \cdot h^{(r_1-r_2)\cdot i}$, which is impossible if p_1 and p_2 are equal.

V. Conclusion

This paper closes the gap that arises from the fact that tenants operating CI services in a cloud environment want a more transparent view while CIPs want to keep their business secrets and to protect other tenants. More specifically, an approach is needed that allows for auditing the virtual to physical resource mapping while preserving corporate confidentiality and possibilities for a later RCA. We presented a way for achieving non-repudiation, evidence gathering for post-mortem analysis, and non-disclosing audits for virtual resource mapping. Tenants are now able to check the fulfillment of some of their requirements without getting knowledge of the concrete cloud set-up. We investigated different solutions like using hash functions or commitment schemes. At this point, we propose to overcome the gap by using a strong computational hiding and perfect statistical binding scheme based on ElGamal commitments. The latter also allows to source out the RCA to a TTP. In addition to that we showed that this auditing information is protected from manipulation and should be strong enough to serve as evidence in a court debate. Finally, it is very important that resulting auditing information can be unveiled to a concrete structure. By applying modern cryptography like commitment schemes it is possible to enhance transparency in cloud scenarios while considering tenant and cloud provider needs. Furthermore, our technique is also applicable for various scenarios in which a CIP has to convince a TIP that something (e.g., physical host affinity, firewall rules, security posture) is equal or different at a given time or at various points in time – without actually disclosing specific values.

Currently, we are implementing the proposed scheme in a Transparency-as-a-Service Framework to get some performance measurements. First preliminary evaluations on a common laptop have shown that ElGamal commitments with near-term security (AES-128 [26]) are in the range of 2 ms and the (anti-)affinity checks are each in the range of 5 ms indicating an acceptable speed for a viable solution.

Acknowledgment

The research presented in this paper has been funded by the European Commission in the context of the Research Framework Program Seven (FP7) project SECCRIT (Grant Agreement No. 312758).

References

[1] A. Haeberlen, "A case for the accountable cloud," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 52–57, Apr. 2010.

[2] R. Ko, B. Lee, and S. Pearson, "Towards achieving accountability, auditability and trust in cloud computing," in *Advances in Computing and Communications*, ser. Communications in Computer and Information Science, A. Abraham, J. Mauri, J. Buford, J. Suzuki, and S. Thampi, Eds. Springer Berlin Heidelberg, 2011, vol. 193, pp. 432–444.

[3] I. Gul, A. ur Rehman, and M. Islam, "Cloud computing security auditing," in *Next Generation Information Technology (ICNIT), 2011 The 2nd International Conference on*, Jun 2011, pp. 143–148.

[4] M. Schöller, R. Bless, F. Pallas, J. Horneber, and P. Smith, "An architectural model for deploying critical infrastructure services in the cloud," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1, Dec 2013, pp. 458–466.

[5] R. Bless, M. Flittner, J. Horneber, D. Hutchison, C. Jung, F. Pallas, M. Schöller, S. N. u. H. Shirazi, S. Simpson, and P. Smith, "Whitepaper 'AF 1.0' SECCRIT Architectural Framework," Feb 2014, https://seccrit.eu/publications/whitepapers.

[6] T. P. Consortium, "Trustworthy Clouds – Privacy and Resilience for Internet-scale Critical Infrastructure," May 2014, http://www.tclouds-project.eu/.

[7] R. B. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao, "NIST Cloud Computing Reference Architecture," in *Proceedings of the 2011 IEEE World Congress on Services*, ser. SERVICES '11. IEEE Computer Society, 2011, pp. 594–596.

[8] R. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. S. Lee, "Trustcloud: A framework for accountability and trust in cloud computing," in *Services (SERVICES), 2011 IEEE World Congress on*, Jul 2011, pp. 584–588.

[9] R. Knode and D. Egan, "A Precis for the CloudTrust Protocol (V2.0)," 2010, https://cloudsecurityalliance.org/research/ctp/.

[10] K. Ronald, "Digital Trust in the Cloud. Liquid Security in Cloudy Places," 2009, http://assets1.csc.com/au/downloads/0610_20_Digital_trust_in_the_cloud.pdf.

[11] H. Xiao, B. Ford, and J. Feigenbaum, "Structural cloud audits that protect private information," in *Proceedings of the 2013 ACM Workshop on Cloud Computing Security Workshop*, ser. CCSW '13. ACM, 2013, pp. 101–112.

[12] D. Bogdanov and A. Kalu, "Pushing back the rain—how to create trustworthy services in the cloud," *ISACA Journal*, no. 3, pp. 49–51, 2013.

[13] K. Yang and X. Jia, "Data storage auditing service in cloud computing: Challenges, methods and opportunities," *World Wide Web*, vol. 15, no. 4, pp. 409–428, Jul. 2012.

[14] C. Wang, S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *Computers, IEEE Transactions on*, vol. 62, no. 2, pp. 362–375, Feb 2013.

[15] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07. ACM, 2007, pp. 598–609.

[16] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. ACM, 2009, pp. 199–212.

[17] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)," RFC 3161 (Proposed Standard), Internet Engineering Task Force, Aug. 2001, updated by RFC 5816.

[18] B. Preneel, "The state of cryptographic hash functions," in *Lectures on Data Security*, ser. Lecture Notes in Computer Science, I. Damgrd, Ed. Springer Berlin Heidelberg, 1999, vol. 1561, pp. 158–182.

[19] R. Damasevicius, G. Ziberkas, V. Stuikys, and J. Toldinas, "Energy consumption of hash functions," *ElAEE*, vol. 18, no. 10, Dec 2012.

[20] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "The Keccak SHA-3 submission," Jan 2011, http://keccak.noekeon.org/.

[21] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology CRYPTO 91*, ser. Lecture Notes in Computer Science, J. Feigenbaum, Ed. Springer Berlin Heidelberg, 1992, vol. 576, pp. 129–140.

[22] M. Abe, R. Cramer, and S. Fehr, "Non-interactive distributed-verifier proofs and proving relations among commitments," in *Advances in Cryptology ASIACRYPT 2002*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, vol. 2501, pp. 206–224.

[23] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *Information Theory, IEEE Transactions on*, vol. 31, no. 4, pp. 469–472, Jul 1985.

[24] Y. Tsiounis and M. Yung, "On the security of elgamal based encryption," in *Public Key Cryptography*, ser. Lecture Notes in Computer Science, H. Imai and Y. Zheng, Eds. Springer Berlin Heidelberg, 1998, vol. 1431, pp. 117–134.

[25] W. Diffie and M. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on*, vol. 22, no. 6, pp. 644–654, Nov 1976.

[26] ENISA, "Algorithms, Key Sizes and Parameters Report," 2013, http://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-sizes-and-parameters-report.