Kent Academic Repository

Full text document (pdf)

Citation for published version

Vu, Quang and Colombo, Maurizio and Asal, Rasool and Sajjad, Ali and El-Moussa, Fadi Ali and Dimitrakos, Theo (2015) Secure Cloud Storage: A Framework for Data Protection as a Service in the Multi-cloud Environment. In: IEEE Conference on Communications and Network Security (CNS), 28-30 September, 2015, Florence, Italy.

DOI

https://doi.org/10.1109/CNS.2015.7346879

Link to record in KAR

http://kar.kent.ac.uk/50845/

Document Version

Author's Accepted Manuscript

Copyright & reuse
Content in the Kent Academic Repository is made available for research purposes. Unless otherwise stated all content is protected by copyright and in the absence of an open licence (eg Creative Commons), permissions for further reuse of content should be sought from the publisher, author or other copyright holder.

Versions of research
The version in the Kent Academic Repository may differ from the final published version.
Users are advised to check **http://kar.kent.ac.uk** for the status of the paper. **Users should always cite the published version of record.**

Enquiries
For any further enquiries regarding the licence status of this document, please contact:
researchsupport@kent.ac.uk

If you believe this document infringes copyright then please contact the KAR admin team with the take-down information provided at **http://kar.kent.ac.uk/contact.html**

KAR Kent Academic Repository

University of Kent

Secure Cloud Storage: A framework for Data Protection as a Service in the multi-cloud environment

Quang Hieu Vu*, Maurizio Colombo*, Rasool Asal*‡, Ali Sajjad†‡, Fadi Ali El-Moussa‡ and Theo Dimitrakos‡
*Etisalat BT Innovation Center (EBTIC)
Khalifa University, United Arab Emirates
Email: {quang.vu, maurizio.colombo}@kustar.ac.ae
†School of Computing, University of Kent
Email: a.sajjad@kent.ac.uk
‡British Telecom
Email: {rasool.asal, fadiali.el-moussa, theo.dimitrakos}
@bt.com

***Abstract*—This paper introduces Secure Cloud Storage (SCS), a framework for Data Protection as a Service (DPaaS) to cloud computing users. Compared to the existing Data Encryption as a Service (DEaaS) such as those provided by Amazon and Google, DPaaS provides more flexibility to protect data in the cloud. In addition to supporting the basic data encryption capability as DEaaS does, DPaaS allows users to define fine-grained access control policies to protect their data. Once data is put under an access control policy, it is automatically encrypted and only if the policy is satisfied, the data could be decrypted and accessed by either the data owner or anyone else specified in the policy. The key idea of the SCS framework is to separate data management from security management in addition to defining a full cycle of data security automation from encryption to decryption. As a proof-of-concept for the design, we implemented a prototype of the SCS framework that works with both BT Cloud Compute platform and Amazon EC2. Experiments on the prototype have proved the efficiency of the SCS framework.**

***Keywords*—Cloud Computing; Data Security, Access Control; Key Management;**

I. Introduction

To address the security concern of cloud computing users, a number of Data Encryption as a Services (DEaaS) have been introduced by various cloud service providers [1], [2]. These services, however, have a couple of major limitations: (1) they often suffer from the problem of 'vendor lock-in', i.e., the encryption service is bounded to a specific cloud service provider or a cloud computing platform and (2) basic security primitives such as access control policies are always out of the their scope, and hence these services are not flexible enough to support data sharing between users due to the need of sharing encryption keys. These limitations force the data owners to trust completely in their cloud service providers and lose control over their data when relying on the cloud for storing, sharing, or accessing it.

To tackle the limitations of the existing DEaaS, in this paper we propose Secure Cloud Storage (SCS), a framework providing Data Protection as a Service (DPaaS) to the users that want to store and share data in a multi-cloud environment. DPaaS is a result of on-going contributions to the ESCUDO-CLOUD project [3] and is being realised as a technical use-case of the project by BT. It aims towards providing data security solutions that guarantee interoperability and enforcement of access restrictions across multiple cloud service providers. In particular, DPaaS provides cloud computing users with not only the basic data encryption capability that is supported by DEaaS but also the capability to define fine-grained access control policies to protect their data. In this way, once data is protected by an access control policy, it is automatically encrypted and only if the policy is satisfied, the data can be decrypted and accessed by either the data owner or anyone else specified in the policy.

The SCS framework is designed to work with any cloud platform and any security solution. Our basic idea is to separate the management process from data encryption process in which data encryption process is open-ended to work with any encryption method and any Key Management Server. For the management process, in addition to allowing users to define access control policies for data, the framework provides automatic provisioning of the protection service at three different levels of granularity: cloud tenant account, virtual machine (VM), and the file/disk inside the VMs. A combination of these three levels provides a full cycle of data security: from tenant account creation, VM provision and data encryption to data decryption, VM de-provision and account termination. Thus, the key challenge addressed by SCS, especially in context of the ESCUDO-CLOUD project, is to offer the key management feature and the policy-based access control feature as a complete life-cycle service.

The rest of this paper is organized as follows. Section II presents related work. Section III discusses requirements and challenges of the framework. Section IV introduces the design of our framework and explains all of its components as well as functionalities in details. Section V shows our

proof-of-concept prototype implementation of the framework. Finally, Section VI concludes the paper.

II. Related Work

Data security in untrusted third parties in general and in the cloud in particular is often provided through access control policies and cryptographic methods [4]. As typical examples, Yu et al. introduced a fine-grained access control for data in untrusted cloud storage [5] based on a combination of attribute-based encryption, proxy re-encryption, and lazy re-encryption whereas Yarlagadda et al. proposed an encryption scheme [6] that integrates Playfair and Vigenere ciphers with the structural aspects of DES and S-DES. To further support data sharing, Kang et al. proposed an Identity-Based Authentication scheme by which the owner can share his encrypted data stored in the cloud [7] while Zhao at el. [8] presented a progressive encryption system based on Elliptic Curve Cryptography that allows the owner to share his encrypted data with other consumers without revealing the plaintext data. Additionally, to address the issue of key loss, Huang et al. presented a key recovery scheme in YiCloud [9]. On the other hand, considering public audit-ability, Wang et al. presented a model on cloud storage for data integrity verification based on Merkle hash tree [10] while Almualla et al. designed a new security architecture with sharing keys [11] that fulfils all security requirements in an environment that supports lawful interception. Finally, Vimercati et al. proposed an approach to address the integrity of join queries in unreliable computational services [12].

The most popular data security service in the cloud is Data Encryption as a Service (DEaaS), which has been introduced in many cloud platforms from commercial ones such as Amazon and Google to open-source ones such as Swift in OpenStack and Cumulus in Nimbus [1], [2], [13]. Besides DEaaS, the authors of [14] introduced a Privacy as a Service, which was implemented by a set of protocols to ensure the security of data in cloud architecture. On the other hand, authors of [15] proposed a Secret Storage as a Service that is designed to securely storing, tracking, and controlling access to digital secrets such as cryptographic keys and hashed passwords. Our proposed framework is different from these services in the sense that we provide a full life-cycle of data security management and maximize the flexibility of users in using the data encryption service. The closest works to ours are [16] and [17]. Specifically, with respect to the work in [16], our service is similar because both services target multi-cloud environments and provide several options for users to customize the services. The difference, however, is that the security service in that work focuses on protecting the VM with firewalls and security tools while our focus is specifically on data encryption. On the other hand, while [17] shares the ideas with our work in providing data encryption as a service, the architecture and design principles of the two works are very different.

III. Requirements and Challenges of the Secure Cloud Storage framework

A. Requirements

Given that Secure Cloud Storage is designed as a framework providing Data Protection as a Service (DPaaS) to cloud computing users, we first discuss requirements for DPaaS. One requirement of our proposed DPaaS is to provide flexibility to cloud computing users by allowing them to define fine-grained access control policies for their data, in addition to the basic data encryption capability. In general, an access control policy provides information of who can access, when to access and how to access data. Another requirement of the DPaaS is to support the full life-cycle of data encryption and decryption, i.e., we expect the DPaaS to provide data decryption for users without any extra step when they want to stop using the service. This feature provides the usability to user and is different from existing DEaaS which always require users to copy data out of the protected zone to receive decrypted data. Finally, DPaaS is required to offer fully automatic installation and configuration of security management, independent of cloud environments and security platforms. When a user wants to protect his data in a particular cloud platform with a specific security solution, all the user needs to do is to specify the cloud platform and the security solution and afterwards everything will be done automatically for the user.

Besides the requirements of DPaaS, there are a couple of specific requirements that we need to address for the framework itself. On the one hand, the framework should work with different cloud platforms. In other words, the framework is expected to provide DPaaS in a multi-cloud environment where users can have accounts from different cloud service providers. Given the popularity of cloud usage, it is not surprised if a user has different cloud accounts and depending on the data type, a specific account will be selected to manage the data (or in the extreme cases, the data can be stored across cloud platforms). On the other hand, the framework should work with different security solutions. Similar to the previous requirement, a user may prefer to employ a specific security solution for a specific type of data, but may use a different security solution for another type of data. In this case, it is expected to support different types of security solutions and provide options for users to select the most suitable one.

In addition to these two separate sets of requirements, we observe that when combining these requirements together, they generate an extra requirement. Basically, in a multi-cloud environment as described above, it is desirable that users are able to employ the same access control policies while transferring the protected/encrypted data across different cloud platforms or locations. It means that the access control policies should be moved with the encrypted data across different cloud service providers easily without the need of re-defining the policies or decrypting and re-encrypting the data. Note that these sets of requirements are general for all types of data. The requirement deliverable of the ESCUDO-CLOUD project [3] provides a more detailed explanation of requirements for DPaaS based on a BT use-

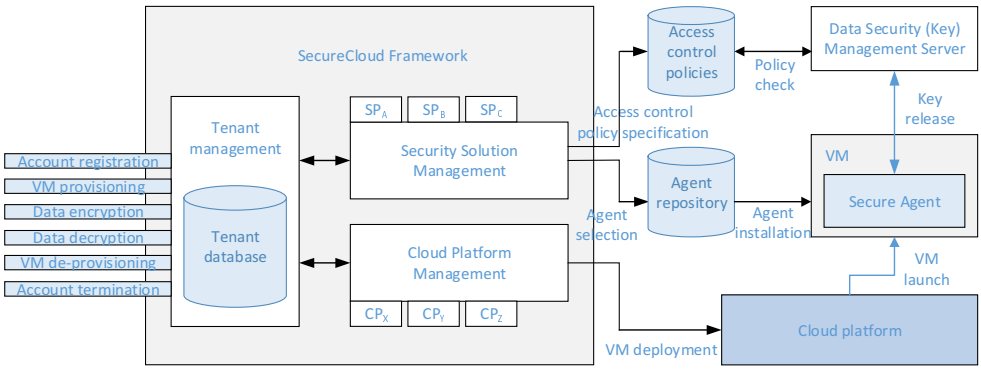


Fig. 1. Architecture of the Secure Cloud Storage framework

case, which is also greater is scope with respect to the cloud storage services being supported by the DPaaS. Thus it covers a superset of the functionalities mentioned here, offering DPaaS for block storage, object storage, and Big Data (HDFS clusters) etc. as well.

B. Challenges

Given the above mentioned requirements, we face three main challenges in building the framework for DPaaS. First, we need to consider differences between different cloud platforms because each cloud platform has a separate way to manage virtual machines as well as their associated resources (e.g., memory, data volumes and network addresses). In particular, given that the platform is to provide service for data protection, the focus is on the differences in virtual machine and data volume management. Second, there is a challenge in communications with outside security servers for policy management and key management. While existing security solutions often follow the Key Management Interoperability Protocol for key exchange between the key management server and the secure agent, there is no standard for policy management, and hence we have to rely on APIs provided by the security solutions. Finally, there is a challenge to maintain the continuation of data access during encryption and decryption. In particular, the challenge is how to avoid data conflict or inconsistency of data during these processes.

IV. Secure Cloud Storage Framework

The architecture of the Secure Cloud Storage framework is shown in Figure 1. In subsequent parts of this section, we will respectively discuss components and functionalities of the framework.

A. Components

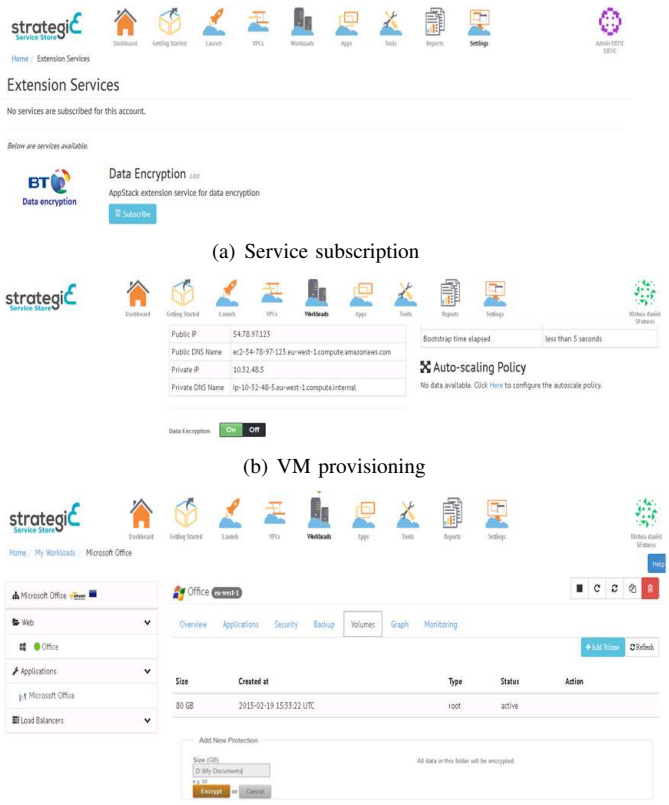
The Secure Cloud Storage framework consists of three main components: Tenant Management (TM), Cloud Platform Management (CPM) and Data Security Management (DSM). TM is the core component of the framework, which is in charge of managing users registering for using the data protection service. In addition, it maintains information of the user cloud platform and security solutions that can be embedded in the data protection service. CPM is the component providing an interface consisting APIs for communications with cloud platforms. For each cloud platform the framework supports, a cloud plug-in implementing APIs of the interface is attached to the component. As shown in Figure 1, there are three cloud plug-ins: CP_X, CP_Y and CP_Z. Among APIs of the interface, some of them are mandatory while others are optional for implementation. For example, it is required to implement the APIs that connects to the cloud platform for VM deployment and VM termination because these operations involve in the security agent installation and uninstallation actions. Similar to the CPM component, DSM is the component providing an interface for communications with the security solution servers and for each security solution the framework supports, a security plug-in is needed. As shown in Figure 1, there are three security solution plug-ins: SP_A, SP_B and SP_C. Basic APIs of this interface include access control policy definition and data encryption/decryption requests.

In addition to the above components, there are three databases supporting the framework: the tenant database, the access control policies database, and the agent repository. Among these three databases, only the first one, which is the tenant database, is located inside the framework. The other two databases are situated outside the framework and have separate interfaces for policy managements. In this way, fine-grained access control policies can be defined and users can customize the policies via the external interfaces without going through the framework. However, by having these two databases outside the framework, we need to also define interfaces to connect them with the DSM component. At the moment, we simply employ a simple interface to set-up default basic access control policies for users and leave users the freedom to add-in or modify the policies later through the external interfaces.

B. Functionalities

The framework allows user interaction via six basic functions. These functions operate on three provisioning levels of the framework: tenant account level, VM level and data level. A brief description of these functions is as follows.

1. Tenant account registration: is used when a user in interested in using the encryption service.



(a) Service subscription

(b) VM provisioning

(c) Data encryption request

Fig. 2. Screen-shots of our implemented prototype

2. VM provisioning: is used when the user wants to launch a new VM or to deploy the security solution in an existing VM. This function is executed to setup and configure the security agent inside the VM.

3. Data encryption: is used when the user wants to encrypt a specific file, folder, or even the whole data volume.

4. Data decryption: is the reverse process of data encryption. This function is used when the user wants to stop protecting the data (i.e., he does not want the data to be encrypted any more). When this function is called, all existing encrypted data will be decrypted.

5. VM de-provisioning: is the reverse process of VM provisioning. When this function is called, security configuration of the VM is first removed. After that, all existing encrypted data associated in this VM is decrypted. Finally, the security agent is uninstalled from the VM.

6. Tenant account termination: is the reverse process of tenant account registration. When this function is called, all VMs that were provisioned for using data protection service in this account will be de-provisioned.

V. Proof-of-concept Prototype

This section shows our proof-of-concept prototype for the Secure Cloud Storage framework where we implemented a Data Protection as a Service working in both BT Cloud Compute platform and Amazon EC2 with choices of security solutions coming from TrendMicro and Vormetric Data Security. This prototype is integrated into a cloud service store and market place that has been developed by BT in the STRATEGIC CIP project[1] building on Appcara AppStack platform [19], which is a platform for fast deployment of cloud computing services in multi-cloud environment. Figure 2 shows main screen-shots of the data provisioning service that include service subscription, VM provisioning and data protection request. In the rest of this section, we will respectively discuss the customer journey of these screen-shots and the result underlining hidden activities inside the framework, which was described earlier in Figure 1.

A. Service subscription

When a user wants to use the Data Protection as a Service to protect their data in a cloud platform, he first needs to subscribe for using the service as shown in the screen-shot Figure 2(a). This request is then processed at the *Tenant Management* component. To fulfil the request, the user needs to set-up two profiles:

1. Cloud profile(s): for each of the cloud platforms that the user has an account and want to have data protection in that account, a cloud profile has to be created. The cloud profile contains the user's credentials that are used to identify the user's account for communications with the cloud platform via APIs. As examples, an Amazon EC2 cloud profile needs the account number, access key and secret key while a Windows Azure cloud profile requires the subscription ID, client private key and client certificate.

2. Security profile(s): for each of the security solutions the user wants to employ to protect his data, a security profile is required. Similar to the cloud profile, the security profile contains the user's credentials that are used to communicate with the security solution servers to manage access control policy or with the key management server to request encryption or decryption keys.

The cloud and security profiles once created are stored in the *Tenant Database*. In cases the user does not have any existing cloud account or security account to create these profiles, the framework will help the user to register with cloud platforms or create accounts with security solutions.

B. VM provisioning

After having registered for using the Data Protection as a Service, the user can provision any Virtual Machine (VM) in registered cloud platforms for data protection. There are basically two cases for VM provisioning: in a new VM or in an existing VM. Figure 2(b) shows VM provisioning in an existing VM. When a VM is chosen for provisioning, the *Cloud Platform Management* component updates information of the VM (e.g., OS, ID, DNS) to the *Tenant*

[1]Please refer to the http://strategic-project.eu/strategic_public_files/deliverables/STRATEGIC_D4.1a_STRATEGIC cloud broker and marketplace_v1.1.pdf file in the deliverable section of [18] for more information about the architecture and capabilities of the service store.

Database. After that, the *Security Solution Management* creates a software update patch and pushes the patch to the VM for downloading, installing and configuring the security agent. As an example, in our framework, Puppet [20] was employed to provide the automatic software update patch.

Basically, to provision a VM with respect to a security solution chosen by the user, the update patch, which is a Puppet recipe in our case, contains: (1) a download link to a suitable security agent installer: this link is created based on the information of the VM recorded earlier and the information of the *Agent Repository Database*, where security agent installers are stored and (2) information of the security server and instructions on how the security agent is installed and configured inside the VM. In addition to create the update patch, based on the security profiles, the *Security Solution Management* component also contacts the key management server as well as any security server of the security solution to confirm the installation and registration of the security agent inside the VM.

C. Data protection request

Figure 2(c) shows that the user can specify any particular folder or file in a VM that has been put in the service for data protection. Additionally, the user can specify the access control policy for the folder if he does not want to employ the default policy.

When a data folder is selected for protection, similar to the case of VM provisioning, both the *Cloud Platform Management* and *Security Solution Management* components are involved in processing the request. In particular, while the *Cloud Platform Management* records the selected folder in the *Tenant Database*, the *Security Solution Management* creates a new software update patch (i.e., the Puppet recipe) to send instructions to the security agent inside the VM. Basically, this update patch only contains: (1) the selected file or folder for protection and the access control policy for that folder and (2) actions that need to be done inside the VM. At the same time, the *Security Solution Management* component also notifies the security server about the data folder that needs protection as well as its access control policy so that the information can be synchronized between the security server and the security agent.

VI. Conclusion

In this paper, we presented a general framework for providing Data Encryption as a Service supporting different security solutions in a multi-cloud environment. Our framework is flexible to support any security solution in any cloud platform through the implementation of specific plug-ins. It provides full automation of data encryption service to users in a complete life-cycle from data encryption to data decryption. A proof-of-concept prototype of the framework was implemented aligned to a subset of the requirements elicited by the BT use-case of ESCUDO-CLOUD project and leveraging a cloud marketplace developed by BT in the STRATEGIC to prove the practicability of our design.

Acknowledgement

The work of some co-authors has been partly supported by the European research and innovation projects STRATEGIC and ESCUDO-CLOUD. STRATEGIC has received funding from the European Unions Competitiveness and Innovation Framework Programme under grant agreement No. 621009 and ESCUDO-CLOUD has received funding from the European Unions Horizon 2020 Research and Innovation Programme under grant agreement No. 644579.

References

[1] "Amazon Encryption Service," http://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html, accessed: 2015-07-07.

[2] "Google Cloud Storage," http://googlecloudplatform.blogspot.co.uk/2013/08/google-cloud-storage-now-provides.html, accessed: 2015-07-07.

[3] "ESCUDO-CLOUD," http://www.escudocloud.eu/index.php.

[4] S. D. C. D. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Encryption policies for regulating access to outsourced data," *ACM Trans. Database Syst.*, vol. 35, no. 2, pp. 12:1–12:46, May 2010.

[5] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing," in *Proceedings of the IEEE INFOCOM*, March 2010, pp. 1–9.

[6] V. K. Yarlagadda and S. Ramanujam, "Data Security in Cloud Computing," *Journal of Computer and Mathematical Sciences*, no. 1, pp. 15–23, 2011.

[7] L. Kang and X. Zhang, "Identity-Based Authentication in Cloud Storage Sharing," in *Proceedings of the International Conference on Multimedia Information Networking and Security (MINES)*, November 2010, pp. 851–855.

[8] G. Zhao, C. Rong, J. Li, F. Zhang, and Y. Tang, "Trusted Data Sharing over Untrusted Cloud Storage Providers," in *Proceedings of the 2nd International Conference on Cloud Computing Technology and Science (CloudCom)*, November 2010, pp. 97–103.

[9] Z. Huang, Q. Li, D. Zheng, K. Chen, and X. Li, "YI Cloud: Improving user privacy with secret key recovery in cloud storage," in *Proceedings of the 6th International Symposium on Service Oriented System Engineering (SOSE)*, December 2011, pp. 268–272.

[10] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel and Distributed System*, pp. 847–859, 2011.

[11] S. A. Almulla and C. Y. Yeun, "New secure storage architecture for cloud computing," in *Proceedings of the International Conference on Future Information Technology (FutureTech)*, 2011, pp. 75–84.

[12] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Integrity for join queries in the cloud," *Cloud Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 187–200, July 2013.

[13] S. Kang, B. Veeravalli, and K. M. M. Aung, "ESPRESSO: An Encryption as a Service for Cloud Storage Systems," in *Proceedings of the International Conference on Autonomous Infrastructure, Management, and Security (AIMS)*, June 2014, pp. 15–28.

[14] W. Itani, A. Kayssi, and A. Chehab, "Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures," in *Proceedings of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, December 2009, pp. 711–716.

[15] A. Sayler and D. Grunwald, "Custos: Increasing Security with Secret Storage as a Service," in *Proceedings of the Conference on Timely Results in Operating Systems (TRIOS)*, October 2014.

[16] J. Daniel, T. Dimitrakos, F. El-Moussa, G. Ducatel, P. Pawar, and A. Sajjad, "Seamless Enablement of Intelligent Protection for Enterprise Cloud Applications through Service Store," in *Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, December 2014, pp. 1021–1026.

[17] P. S. Pawar, A. Sajjad, T. Dimitrakos, and D. W. Chadwick, "Security-as-a-Service in Multi-cloud and Federated Cloud Environments," in *Proceedings of the 9th IFIP International Conference on Trust Management (IFIPTM)*, May 2015.

[18] "STRATEGIC CIP," http://www.strategic-project.eu/.

[19] "Appcara," http://www.appcara.com/.

[20] "Puppet," https://puppetlabs.com/.