Future Generation Computer Systems 57 (2016) 24–41

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Cloud computing adoption framework: A security framework for business clouds

Victor Chang [a,*], Yen-Hung Kuo [b,*], Muthu Ramachandran [a]

[a] School of Computing, Creative Technologies and Engineering, Leeds Beckett University, Leeds, UK

[b] Data Analytics Technology & Applications, Institute for Information Industry, Taiwan, ROC

HIGHLIGHTS

- We demonstrate CCAF multi-layered security.
- We explain the mappings between CCAF multi-layered architecture and core technologies
- We performed penetration testing and SQL injection on CCAF multi-layered security.
- Results and analysis by CCAF are better than those produced by the other tools.
- CCAF multi-layered security blends with policy, services and business activities.

ARTICLE INFO

Article history:
Received 11 July 2015
Received in revised form
9 September 2015
Accepted 27 September 2015
Available online 19 October 2015

Keywords:
Cloud computing adoption framework (CCAF)
OpenStack
CCAF multi-layered security
Security for business clouds

ABSTRACT

This article presents a cloud computing adoption framework (CCAF) security suitable for business clouds. CCAF multilayered security is based on the development and integration of three major security technologies: firewall, identity management, and encryption based on the development of enterprise file sync and share technologies. This article presents the vision, related works, and views on security framework. Core technologies have been explained in detail, and experiments were designed to demonstrate the robustness of the CCAF multilayered security. In penetration testing, CCAF multilayered security could detect and block 99.95% viruses and trojans, and could achieve $\geq$85% of blocking for 100 h of continuous attack. Detection and blocking took $<$0.012 s/trojan or virus. A full CCAF multilayered security protection could block all SQL (structured query language) injection, providing real protection to data. CCAF multilayered security did not report any false alarm. All *F*-measures for CCAF test results were $\geq$99.75%. The mechanism of blending of CCAF multilayered security with policy, real services, and business activities has been illustrated. Research contributions have been justified and CCAF multilayered security can be beneficial for volume, velocity, and veracity of big data services operated in the cloud.

1. Introduction

Security, trust, and privacy always remain challenges for organizations that adopt cloud computing and big data. Although there are demands for businesses to move their data to the cloud and centralize management for data centers, services and applications are designed to reduce cost and increase operational efficiency. System design and deployment based on current security practices should be simultaneously enforced to ensure compliance of all data and services with up-to-date patches and policies. A risk-based approach to the development of a security program that recognizes (and funds) appropriate controls will ensure protection of all users and confidentiality, integrity, and availability of data.

Some researchers have adopted a framework approach that allows organizations to follow guidelines, policies, and standards. For example, Zhang et al. [1] propose a usage-based security framework (UBSF), which can consolidate guidelines and policies with their framework, architecture, and digital certificates. Takabi et al. [2] describe a comprehensive security framework via a model that explains the method of working with different service integrators and service providers. Zia and Zomaya [3] present a wireless sensor network model with algorithms and a software engineering approach. All these frameworks have recommendations

* Corresponding authors.
E-mail addresses: V.I.Chang@leedsbeckett.ac.uk (V. Chang), keh@iii.org.tw (Y.-H. Kuo).

http://dx.doi.org/10.1016/j.future.2015.09.031
0167-739X/

on guidelines to use. However, there are no details on the actual use of these proposals and also no clear evidence of adoption of these proposals to business clouds, whose requirements include ease of use, adaptability, best practice compliant, and support by large-scale experiments such as penetration testing to validate robustness of such proposals [4,5]. Indeed, without such a clear "line of sight" between conception and implementation, such frameworks are unlikely to achieve operational status.

The cloud computing adoption framework (CCAF) has been developed to meet the requirements of business clouds and ensure that all implementations and service deliveries overcome all the technical challenges. Real-life case studies show how different cloud computing designs and their development and service delivery overcome both technical and organizational challenges. In the first example, CCAF was the framework used to develop cloud storage and bioinformatics solutions for biomedical scientists based in the United Kingdom at Guy's Hospital and King's College London [6]. This framework ensured the deliveries of storage services to back up thousands of terabyte-sized medical data. Bioinformatics services can simulate DNAs, proteins, genes, tumors, and organs of the human body. The use of this security is limited to authentication, encryption, and users with authorized access. In the second example, CCAF is used to provide guidelines for financial modeling, so that the best practice and call prices can be computed with respect to the change of risks. Advanced computational techniques have been used to calculate risks and market volatility [7]. Security is limited to password authentication and users with authorized access and biometrics checks for financial simulations. In the third example, investigations of hacking methods have been studied and made as part of prototype requirements. User requirement and literature review have identified factors for a successful implementation. All the collected and synthesized data have been instrumental in the development of CCAF Version 1.1, which emphasizes on the security policies, recommendations, techniques, and technologies to be updated in the framework [8]. In the aforementioned examples, a more comprehensive cloud security solution is required to ensure robustness and resistance of the services to attack, hacking, and unauthorized attempts to gain access. More experiments and simulations are required to validate the robustness and effectiveness of the proposed security framework. This motivates us to consolidate our CCAF framework by providing a holistic approach involved with service integration, OpenStack security, and multilayered security to enhance security for business clouds. An integrated security framework is proposed for business clouds to have the multilayered security in place and the large-scale penetration testing and experiments to validate the robustness and effectiveness of our approach. All these proofs of concepts and lessons learned are important to big data in the cloud as follows. First, it ensures that all the cloud services are safe and secure, including the incoming and outgoing data of the organizational data centers hosted on hundreds and thousands of virtual machines (VMs). Second, it ensures that large amount of data and large data sets can be processed and analyzed safely in the cloud, which also explains the necessity of large-scale penetration testing to validate the framework.

The organization of this article is as follows: Section 2 presents the literature for security. Section 3 describes our core security technology for enterprise file sync and share (EFSS), including the architecture and layered components. Section 4 explains the multilayered approach with core technologies and results from large-scale experiments for penetration testing, SQL (structured query language) injection, and data scanning. Section 5 illustrates topics of discussion, and Section 6 summarizes conclusion and future work.

2. Literature

The following are the different types of security frameworks proposed so far. Zhang et al. [1] propose their UBSF for collaborative computing systems. They explain their motivation, techniques used, architecture, and conditions for experiments. The decision on the use of UBSF is made based on subjects, objects, authorization, obligations, and conditions. With support from literature and hypotheses, they explain their model's mechanism of work in collaborative ways. The usage-based authorization architecture uses sensors, directory service, policy decision point (PDP), and usage monitor (UM) to functions. Steps have been described to justify the effective function of UBSF. In order to assist UBSF, Zhang et al. [1] include a prototype system architecture. They use OpenLDAP and OpenSSL to enforce security. They have three types of digital certificates: user, attribute repository (AR), and resource provider (RP). They explain the use of these certificates in their workflow of security processes. They also adopt extensible access control markup language (XACML) to enforce policy specification, which aligns with the UBSF approach for security. Ko et al. [9] investigate trust for cloud computing and propose a TrustCloud framework that focused on accountability. It has three layers: (1) system layer that covers all the underlying hardware and platform; (2) data layer that contains the data for the work; and (3) workflow layer that uses workflow to execute all the services and requests. In addition, two nonfunctional layers are associated with these three layers. The first layer is laws and regulations, which ensures all services follow the legal requirements of the country in which the service was delivered. The second layer is policies, which are the consolidated service-level agreements and the best practice approach. This framework is considered as a conceptual framework focused on the recommendations and best practice, as they do not include quantitative analyses, computational demonstrations, and case studies. Pal et al. [10] present their cloud security that has emphasized on the architecture and steps of interactions between different services. They explain the role of each major user, their agents, and all the 15 steps involved. They use unified modeling language (UML) diagram to justify their approach and architecture to explain the relationship between the user, provider, proxy server, user agent, and provider agent. They present two algorithms and experimental results. They validate their approach using "trust value updation". However, their assumption is based on the probabilities of 0.8 and 0.2 of having a trusted and nontrusted user, respectively. There is no evidence supporting this, and they do not use any reference or survey to justify their research. This also depends on the sample size, demographics, and the country in which the research was conducted. The National Institute of Standards and Technology (NIST) [11] framework provides a common language for establishing cybersecurity. The core NIST framework provides a set of activities to identify, protect, detect, respond, and recover without more specific examples and case studies implementing a full-security solution. However, our work on CCAF extends to detailed activities and implementation on security for cloud computing and big data.

All these examples have security framework. However, the proposals described above do not demonstrate their contributions to business clouds. In other words, when businesses adopt cloud computing solution, they should be able to provide architecture, approaches for their framework, and steps and experiments to support the robustness and validity of the framework. Our proposal on CCAF provides details on core technologies in Section 3, and the theoretical framework mapping of core technologies is shown in Section 4 with experimental results validating our framework. Key topics, including security policy, business and security alignment, framework and core technology integration, relation of the big data in cloud, and overall contributions with limitation, are discussed

in Section 5. Finally, research conclusion and future work are discussed in Section 6.

3. Core technologies

Prior to the introduction of CCAF, this section uses a concrete instance of the CCAF to be an example to explain CCAF core technologies and implementations. In order to meet the requirements of moving big data in a semipublic business cloud, an enterprise cloud storage application – the semipublic EFSS service – is chosen to be the CCAF instance to explain the mechanism of protection of enormous enterprise files (a kind of unstructured big data) by CCAF in a business cloud environment.

In order to provide enterprises with the convenient cloud file sync and share service while taking enterprise concerns, such as security, compliance, and regulation, into consideration, the service was been deployed by either on-premise or hybrid cloud model to target high-value EFSS market [12–14]. Existing EFSS systems focus on system security and manageability, which encrypt data on transfer and at rest, and also support system audit trail. As part of the core technologies of the CCAF framework, important EFSS security issues should be well addressed, particularly for businesses with critical data services. The following are the EFSS security issues:

(1) **Employee Privacy**: In order to prevent data leak, enterprise data are usually encrypted in existing EFSS systems. However, most existing EFSS systems only use single master key to encrypt entire data space, which can prevent enterprise data leaks from outside but not from inside. For example, an EFSS system administrator (information technology (IT) or management information system (MIS) in the enterprise) can spy enterprise-sensitive data by self-granted authority.
(2) **Share Link**: The share link is widely adopted to share data with business partners who do not have an EFSS system account. From enterprise's perspective, the share link is convenient, but not secure, as it involves new security loophole that might be used to leak data to unauthorized domain without leaving enough audit trail [15]
(3) **Cloud File Synchronization**: The nature of the cloud file synchronization is a security loophole to enterprises. It synchronizes shared and collaborative enterprise data from a managed EFSS service to employees' endpoint devices, and enterprises then have less/no control on the synchronized enterprise data. The synchronized enterprise data can then be distributed from the endpoint devices to other unauthorized domains via e-mail, USB disk, and other communication interfaces available on the endpoint devices.
(4) **Enterprise Directory Integration**: In order to enable single sign-on (SSO) in the enterprise, most existing EFSS systems, via direct network connection, integrate its authentication with existing enterprise directory (e.g., active directory (AD) and lightweight directory authentication protocol (LDAP)). It also introduces two new security issues. First, the EFSS system can access employees' profiles available in enterprise directories. Second, the EFSS system can log employees' credential information during authentication, as their usernames and passwords pass through the EFSS system to enterprise directory to conduct the actual authentication. Both cases provide EFSS system with chances to obtain authorized information.

An integrated security approach, which integrates several key components to form a scalable secure EFSS system to address the enterprises' concerns by leveraging the on-premise OpenStack infrastructure [16], is introduced in the following sections. EFSS has been integrated with the CCAF framework as an overarching model for cloud security in businesses.

Architecture and Design of the Integrated Security Approach

The key components of the secure EFSS system are virtual appliances that can be provisioned and run on the OpenStack compute service, Nova, and the data (e.g., metadata in database and uploaded enterprise files in user storage space) generated by these components are stored in the OpenStack storage services: Cinder and Swift for block storage and object storage, respectively, where the OpenStack storage services are managed by the storage system controlled by the enterprise. The separation of the compute and storage makes the Secure EFSS system is scalable and more secure than existing EFSS systems.

All key components/virtual appliances, including load balancer, firewall, virtual file system (VFS) service, directory service, log service, message queue (MQ) service, and database service, are provisioned from an integrated image stored in the OpenStack VM image management service, Glance, to form the secure EFSS system as shown in Fig. 1. The VFS service of the proposed secure EFSS system provides clients with a representational state transfer (REST) application programming interfaces (API) set to manipulate a directory structure and files of a VFS. The back end of the VFS is the database service, which stores metadata to represent the directory structure and the nodes' attributes of the directory structure. Enterprise files uploaded by clients are leaf nodes of the directory structure, and they have a node attribute to indicate encrypted objects in the Swift. Because no data are stored in the VFS service, it can dynamically provision more VFS VMs to fulfill the increasing demand of EFSS system in a scalable and distributed manner.

The interactions of the REST key components and their benefit to security and scalability are shown in Fig. 1. The load balancer is located in the demilitarized zone (DMZ), which dynamically distributes every request made by the sandbox-based cloud file sync app to one of the VFS VMs for handling the requests according to the loads on it for maximizing the overall VFS service performance. Moreover, a firewall is located between the load balancer and the VFS service to form a secure deployment scheme, which defends external direct network attacks for internal services. Once the VFS service receives a request, it first sends the authentication information of the request to the directory service to check the identity and authority of the request. Subsequently, the VFS service handles the request and logs related audit trails to the log service. During VFS service, the overall service status, including concurrent serving requests, CPU, and memory usages, is reported to MQ service, which actively calls OpenStack compute service API to provision and de-provision VFS VM to handle the peak and idle situations of the secure EFSS system.

Designs of key components of the proposed integrated security approach are introduced in the following subsections, which include user storage space modeling, distinct share link, zero-knowledge cloud scale file sharing system, sandbox-based cloud file synchronization, out-of-band authentication, and a NoSQL adoption consideration.

3.1. User storage space modeling

In order to ensure scalability of the system, metadata generated by key components, particularly by the VFS service, are stored in a MongoDB cluster, which is a document-based NoSQL database that increases scalability by sacrificing the relationship between documents [17,18]. Furthermore, user accounts are used as database sharding key to shard the user storage space to mitigate the MongoDB scalability limitation, which affects database performance when too many documents are inserted into a data collection process. Then, every user has a directory structure formed by metadata stored in his/her own data collection, which physically isolates the user's metadata. Therefore, every user's

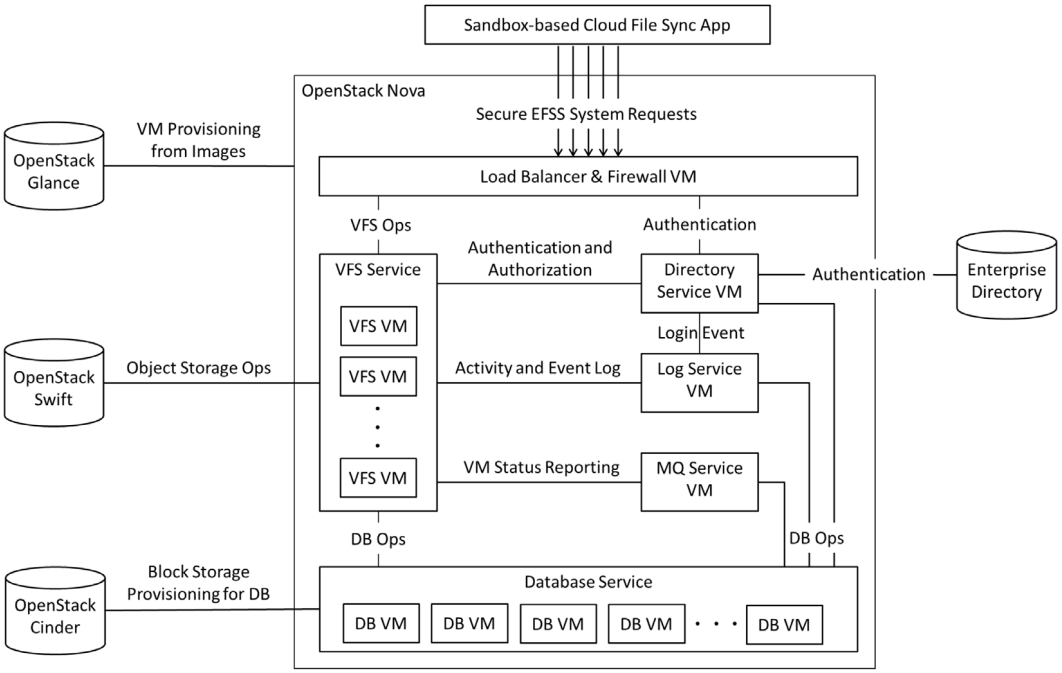


Fig. 1. Secure EFSS system architecture in the OpenStack.

metadata can be protected by owner's encryption key without leaking of sensitive metadata when database service is hacked.

3.2. Distinct share link

Sharing of cloud files between an enterprise and external business partners can be easily performed using a share link. However, it is insecure as web crawlers can simply download it by scanning e-mails and social network accounts [19]. This problem can be overcome by a distinct share link that secures sharing of cloud files and builds a secure sharing relationship between two user storage spaces, which are isolated data collections sharded from the user storage space [20].

The distinct share link is an additional layer associated with permissions, identities, and access conditions. It encapsulates a share link, decreases diffusibility of the share link, adds traceability as well as controllability to the share link, and then sends it to recipients with attached identities. A recipient trying to gain access to a distinct share link has to input his/her identities for access condition check. After passing all checks, the distinct share link layer prepares an ephemeral representation to access the share link. Because every distinct share link access requires identity (traceability) that is controlled by permissions and access conditions (controllability), its diffusibility and convenience are decreased.

In order to overcome this problem, a recipient-defined identity function is introduced, which allows the recipient to define own identity (e.g., password and secret code) for each received distinct share link. Hence, the recipient can define easy-to-remember identities to received distinct share links. It also resolves the problem of share link identity management, that is, share links with different identities (e.g., passwords) are too much information to be remembered by a single recipient.

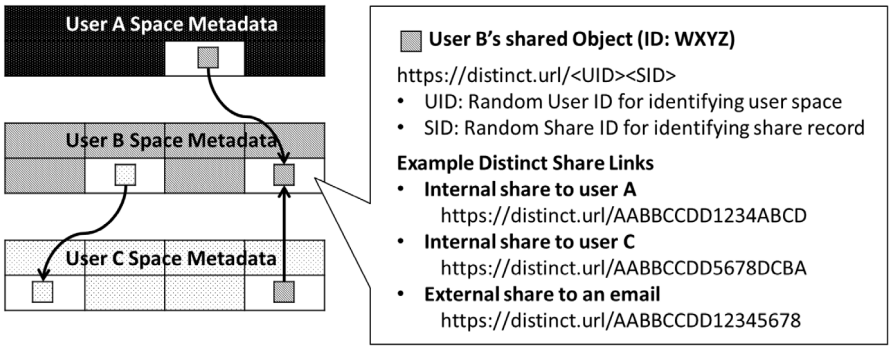
In addition to the aforementioned benefits, the distinct share link further supports sandbox feature for creating a collaborative workspace with external business partners, where every individual has own permission, and any inappropriate action (e.g., delete all files) can be performed only in a specific shared folder, but not in the entire personal user storage space. It is worth noting that with cloud storage, any performed inappropriate action can be undone, for example, previous cloud file version and deleted shared files can be recovered from file change history and recycle bin, respectively.

The distinct share link is also used to implement the internal cloud file sharing in the proposed secure EFSS by building a secure sharing relationship between two isolated user storage spaces (data collections). Fig. 2 depicts the mechanism of building the internal cloud file sharing by distinct share link. The top-left part of the figure presents two internal cloud file sharing relationships: (1) users A and C receive an object shared from user B and (2) user B receives an object shared from user C. The callout box shown in the top-right part of the figure further explains the sharing relationships associated with the object (ID: WXYZ) in the former sharing relationship. It shows that three distinct share links are used to share the object (ID: WXYZ) with the recipients: user A, user B, and an e-mail address. Moreover, all the three distinct share links can be parsed into three parts: (1) the service endpoint (e.g., https://distinct.url/); (2) the UID part for identifying user storage space, that is, location of the data collection; and (3) the SID part for identifying the metadata (table in Fig. 2), which contains information about the sharing relationship in the located data collection. After retrieving the metadata, the "ObjectID" can be used to refer to either a shared object or share link point to a shared object.

The aforementioned external and internal sharing cases have similar distinct share link creation and access processes. The key difference between the two cases is the identity delivery process. In external sharing, the identity is first defined by the distinct share link creator, subsequently it is delivered by oral or identity itself (e.g., e-mail), and finally it might be redefined by the recipient. The internal sharing automatically performs similar processes by system with the following steps: (1) generates random keys as identities in the sharing source, (2) encrypts generated identities by a recipient's public key, and (3) delivers the encrypted identities from the sharing source to the sharing destination for further use.

3.3. Zero-knowledge cloud scale file sharing system

All cloud files are stream-encrypted by different random symmetric keys and stored in the Swift object storage when



Collection Name (UID): AABBCCDD

SID	ObjectID	Permission	Identity	Access Condition
1234ABCD	WXYZ	Read Only	User A Identity + Secrets for User A Space Metadata	Must from: 127.0.0.1 Expired within 20 hours
5678DCBA	WXYZ	Read & Write	User C Identity + Secrets for User C Space Metadata	Must from: 127.0.0.1 No usage limit
12345678	WXYZ	Read Only	Email Identify + Secrets for email owner	Allow from Internet Download up to 20 times

The shared object ID (ObjectID) can be then used to refer to a real record in user B space metadata.

Fig. 2. Example of a distinct share link.

uploading to the proposed secure EFSS system. A zero-knowledge cloud scale file sharing system is introduced herein, which leverages introduced stacks, including user storage space modeling technique and distinct share link. This system also introduces a new key cascading model [21,22], which manages the random symmetric keys crossing loosely coupled domains and further protects cloud files from internal/external unauthorized access attempts.

In order to develop a zero-knowledge cloud scale file sharing system, the following key enterprise security concerns and scalability and manageability issues of the modern cloud system are considered:

(1) **Zero-knowledge System**: Zero-knowledge means that the system knows nothing about the information provided by users, which can be contents of cloud files, metadata of user storage spaces, and encryption keys, including passwords that protect cloud files and user storage spaces. Finally, yet importantly, the system must apply to the entire life cycle of data. Therefore, the zero-knowledge cloud scale file sharing system is formed by stacking multiple distinct domains with own services, and each of the distinct domain serves as an additional complementary security layer to its preceding domain. As shown in Figs. 3 and 4, these domains represent stacking relationships of personal cloud file and shard cloud file situations, respectively. In Fig. 3, the lowest OpenStack domain only provides cloud file storage capability; subsequently, the succeeding VFS service domain servers cloud file metadata management functions and add additional stream cloud file and metadata encryption capabilities. Finally, the topmost directory service domain incorporates the user domain to protect the encryption keys used in cloud file and metadata protections. It is worth noting in the aforementioned stacking relationship that the user domain is not in cyberspace, rather it is in the user's brain, that is, only the user knows his/her own password, thereby rendering an added advantage to the zero-knowledge system. Similarly, Fig. 4 further establishes a sibling stacking relationship with Fig. 3 to isolate the users' personal VFS service domains (see distinct personal databases of User1 and User2 in Fig. 4). This further makes the zero-knowledge system work in a multitenant cloud environment. As the sibling stacking relationship is a one-way relationship, users are needless to worry about their cloud files when another user's storage space was conquered, that is, a lower layer cannot use existing information to derive upper-layer information. Finally, this system also guarantees users have no knowledge about each other.

(2) **Security Compliance**: In addition to data encryption, enterprise security compliance must also be capable of password history management. For example, when an employee opts to change his/her password every quarter, the new password cannot be similar to any passwords used in the previous three quarters. The zero-knowledge cloud scale file sharing system abstracts a directory service domain to fulfill the requirement of password history management (Figs. 3 and 4). For every password change, the directory service domain will automatically adjust its internal relationship to maintain consistency with the stacking relationship without affecting the lower domains. In other words, the zero-knowledge cloud scale file sharing system is also a security compliance friendly system.

(3) **Atomic Objects and Systems**: Considering the complex cloud environment, making all objects atomic is a good idea. In the zero-knowledge cloud scale file sharing system, every service in the domains and every encrypted cloud file is an atomic unit, which can be managed (e.g., create, update, and delete) regardless of their dependencies with other atomic units. In order to achieve atomicity, the data are separated from computation logic. As mentioned earlier, metadata of a user are aggregated to form a personal database, and because of choosing MongoDB, the personal database is physically a set of manageable files stored in block storage. Furthermore, cloud files are themselves atomic and can be directly stored in object storage. However, managing an encrypted cloud file as an atomic unit is difficult, as the encryption key of the cloud file has to be managed as well. Therefore, in order to achieve atomicity, it encrypts the encryption key of the cloud file, combines the encrypted encryption key and the encrypted cloud file to form a manageable encrypted object to replace the original plaintext cloud file, and allows the encryption key of the encrypted cloud file to be managed by the VFS service domain. As shown in the dotted-line block in Figs. 3 and 4,

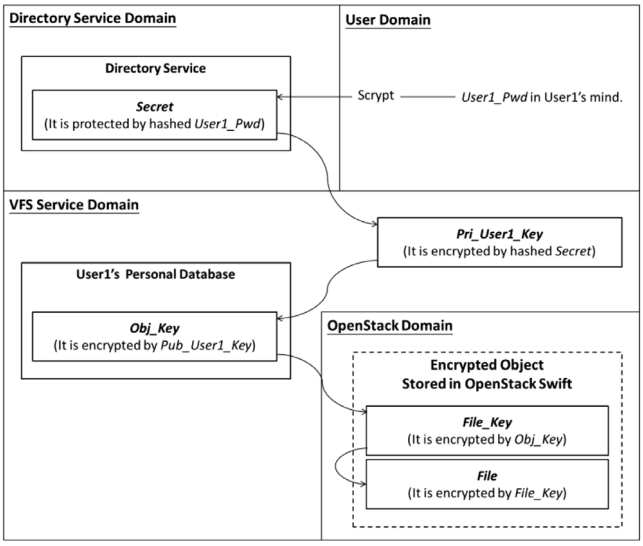


Fig. 3. Personal file encryption key management concept.

the encrypted cloud file (*File*) and its encryption key (*File_Key*) are combined as an encrypted object stored in the OpenStack Swift object storage. The encrypted object thus becomes an atomic unit, which can be migrated, backed up, and managed in a cloud manner. The remaining parts are computation logics that can be categorized by domains (e.g., Directory Service and VFS service domains have their own computation logics). Different types of computation logics can be implemented in VMs or container images that can be managed by the OpenStack Glance image service as atomic units and hosted on the OpenStack Nova cloud computing environment.

(4) **API Integration**: All domains in the zero-knowledge cloud scale file sharing system are integrated by REST API, which provides flexible services in domains by developing and growing themselves. With such an integration, every domain need to focus only on satisfying contracts and/or service level agreement (SLA) with other domains (e.g., maintaining 99.99% service availability or guaranteeing API calling 300 times per minute). In addition to interdomain integration, the VFS service domain also uses REST API to implement the integration of distinct user storage spaces. Particularly, it uses distinct share link to securely share cloud files from one user storage space to that of the other user (see the dotted-line arrow in Fig. 4). It forms a loosely coupled sharing relationship, which is particularly suitable for sharing cloud files.

Detailed designs and implementation of the zero-knowledge cloud scale file sharing system will be discussed below. The core technology that integrates all the aforementioned key considerations is a key cascading model. Figs. 3 and 4 are used as examples to assist the follow-up explanations.

Fig. 3 is the key management concept of the personal storage space, where the solid-line arrows indicate encryption/decryption directions, which are shown by the following two examples. In the first example (a symmetric key case), "*Secret*" pointing to "*Pri_User1_Key*" has two meanings: (1) the "*Pri_User1_Key*" is encrypted by the (hashed) "*Secret*" and (2) the "*Pri_User1_Key*" is capable of being decrypted by the (hashed) "*Secret*". In this example, the "*Pri_User1_Key*" is the private key of User1 and the "*Secret*" could be any information managed by the directory service. It is important to note that keys in the key cascading model could point to the "*Secret*", which is hashed before being used to encrypt other keys. In another example (an asymmetric key case), the "*Pri_User1_Key*" pointing to the "*Obj_Key*" has only one meaning, that is, the "*Obj_Key*" is capable of being decrypted by the

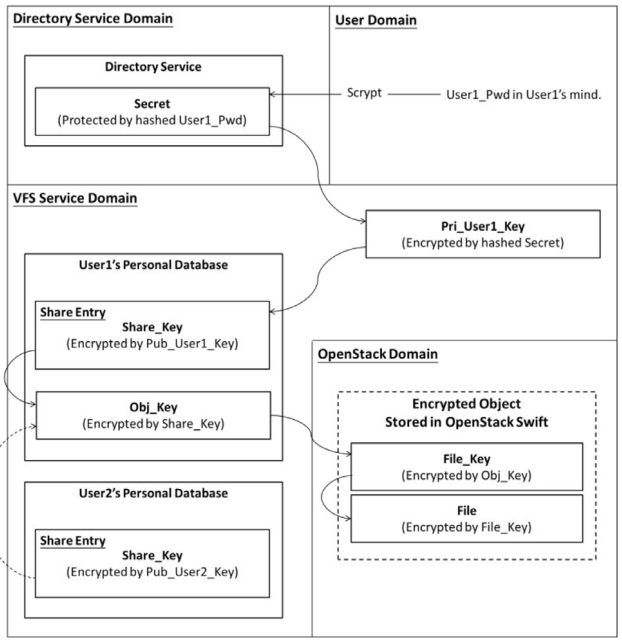


Fig. 4. Shared file encryption key management concept.

"*Pri_User1_Key*", because the "*Obj_Key*" is originally encrypted by the public key of User1. Fig. 4 is the key management concept of the shared storage space, where the solid- and dotted-line arrows have the same definition with the solid-line arrows in Fig. 3.

As shown in Fig. 3, the key cascading relationship of the personal storage space crosses four distinct domains, starting from the user domain, passes through the directory service and VFS service domains, and ends in the OpenStack domain. Starting at the tail of the key cascading relationship, in order to obtain the plaintext "*File*", the random symmetric key, "*File_Key*" is required to decrypt the encrypted "*File*". The "*File_Key*" can be further decrypted by another random symmetric key "*Obj_Key*", which is encrypted by User1's public key in the VFS service domain. In order to prevent the "*File_Key*" from exposing risk during sharing or changing ownership, User1's public key does not directly encrypt the "*File_Key*", but rather encrypts the "*Obj_Key*", which protects the "*File_Key*". Thus, User1's public key can protect the "*File_Key*" indirectly, and the plaintext "*File_Key*" will be required only when trying to gain access to the "*File*". This property can be used to combine the encrypted "*File_Key*" and the encrypted "*File*" to form an encrypted object (atomic unit) to be stored in the Swift object storage, which is also beneficial for object migration, replication, and management.

Reconsidering the key cascading relationship, User1's private key, "*Pri_User1_Key*" is the entry of the VFS service domain, which is formed by metadata stored in the user's personal data collection. As the entire VFS service domain may be conquered, another key, which is not available in the VFS service domain, must be chosen to protect "*Pri_User1_Key*". Because of the nature of authentication and authorization always take place before accessing the VFS service domain, the implementation chooses the "*Secret*" in the directory service domain to protect "*Pri_User1_Key.*" The best practice of choosing suitable information from the directory service domain to be the "*Secret*" is to choose the information whose changing frequency conforms to enterprise compliances, because "*Pri_User1_Key*" has to be reencrypted together with the changing of the "*Secret*". Finally, User1's password is used to protect the "*Secret*". In the key cascading model, User1's password is the only information not available in the cyberspace, as the hash of the password has already qualified to be used to protect the

The key managementalgorithm for sharing a personal file:

STEP 1. Obtain *Obj_Key*:
// The *User_Pwd* is the user's password.
Secret←*User_Pwd* (encrypted *Secret*);
// The *Pri_User_Key* is the user's private key.
Pri_User_Key←*Secret* (encrypted *Pri_User_Key*);
Obj_Key←*Pri_User_Key* (encrypted *Obj_Key*);

STEP 2. Obtain *Share_Key*:
IF *Share_Key* for the share does not exist THEN
// Generate a new Share_Key.
Share_Key←Share_Key_Gen();

STEP 3. Encrypt *Obj_Key* by *Share_Key*:
encrypted*Obj_Key*←*Share_Key* (*Obj_Key*);
store the encrypted *Obj_Key* into personal database to replace the encrypted *Obj_Key* in SETP 1;

STEP 4. Encrypt *Share_Key* by *Public Key*:
// The *Pub_User_Key* is the user public key.
encrypted*Share_Key*←*Pub_User_Key* (*Share_Key*);
SWITCH (identity)
CASE 'owner':
store the encrypted *Share_Key* into personal database with the encrypted *Obj_Key* as the share entry;
CASE 'recipient':
store the encrypted *Share_Key* into personal database as the share entry;
END

Fig. 5. Key management algorithm for sharing a personal file.

"*Secret*". However, in our implementation, the password is hashed by the Scrypt [23], which generates different hash values every time, demanding us to choose another suitable "*Secret*" that is associated with the password.

In contrast to the key management concept of the personal storage space, the key management concept of the shared storage space, as shown in Fig. 4, adds another key cascading relationship between sharing individuals' private keys and object key of the shared object. In order to add another key cascading relationship among sharing individuals' data collections, a symmetric key "*Share_Key*" is inserted between the individuals' private keys (e.g., "*Pri_User1_Key*" and User2's private key) and the "*Obj_Key*". As shown by the dotted-line arrow in Fig. 4, User2 is can use the "*Share_Key*" as part of the identity through the distinct share link to gain access to User1's shared objects (e.g., "*File*").

In order to represent the aforementioned explanations formally, the key management algorithms used for sharing a personal file and stopping a sharing process are described in Figs. 5 and 6, respectively.

3.4. Sandbox-based cloud file synchronization

The EFSS service is a convenient way of collaboration in an enterprise. However, most IT and MIS do not prefer to provide EFSS service because of the fact that, once a cloud file is shared and synchronized with an employee's client device, it is no longer under the control of the enterprise. In order to provide an integrated secure EFSS approach, a **Sandbox-based Cloud File Sync App** is proposed [24,25]. In contrast to the traditional cloud file synchronization tool, the sandbox-based cloud file sync app creates a managed sandbox in the operating system's (O/S) file and a local sync folder inside the sandbox, and then synchronizes the cloud file between the local sync folder and user storage space in the secure EFSS system. As only authorized file system operations can be performed in the sandbox, the O/S and applications executed on top of the O/S are unable to read or update the sandbox. This property makes the sandbox-based cloud file sync app capable of actively

The key managementalgorithm for stopping a share:

STEP 1. Obtain *Obj_Key*:
Secret←*User_Pwd* (encrypted *Secret*);
Pri_User_Key←*Secret* (encrypted *Pri_User_Key*);
Share_Key←*Pri_User_Key* (encrypted *Share_Key*);
Obj_Key←*Share_Key* (encrypted *Obj_Key*);

STEP 2. Encrypt*Obj_Key* by *Pub_User_Key*:
encrypted*Obj_Key*←*Pub_User_Key* (*Obj_Key*);
store the encrypted *Obj_Key* into personal database to replace the encrypted *Obj_Key* in SETP 1;

STEP 3. Remove *Share_Key*:
remove corresponding *Share_Key* from owner's personal database;
/*
Note that recipients'*Share_Key* are going to be removed asynchronously when trying to access the missing share entry.
*/

Fig. 6. Key management algorithm for stopping a sharing process.

preventing cloud files in an employee's client device from leaking. The dotted-line area in Fig. 7 is the enterprise domain, which is extended from the original secure EFSS system to every employee's client device.

The core of the sandbox-based cloud file sync app is the I/O monitoring and filtering module, which is a file system filter as shown in Fig. 7. These refer to isolated object list and I/O filtering rules to monitor and filter file system operations trying to apply to files and folders in the sandbox. A list of file and folder paths is available in the isolated object list to form the sandbox isolated file system portion of the file system. Moreover, this list is dynamically updated whenever the object sync logic synchronizes files and folders between the local sync folder and secure EFSS system. In order to make the sandbox-based cloud file sync app an enterprise compliance facilitating tool, the I/O filtering rules provide the IT and MIS of an enterprise the capability to update its I/O filtering rules, thereby controlling the local synchronized files through controls made by the secure EFSS system.

3.5. Out-of-band authentication

As mentioned earlier, directly integrating EFSS system with enterprise directory to enable SSO is insecure as sensitive information may be accessed or logged by the EFSS system. In order to overcome this problem, an out-of-band authentication method is introduced as part of the integrated security approach, whose sequence diagram is shown in Fig. 8. In order to achieve security, the individuals involved in the authentication process are deployed in four different networks: (1) the client can gain access to the service from either Internet or Intranet; (2) the service is deployed in either DMZ or the enterprise's private local area network (LAN) based on the purpose of the service; (3) the directory service has to be deployed in DMZ to handle authentication requests from both Internet and Intranet; and (4) behind the firewall, the **Enterprise Directory** (e.g., AD or LDAP) is protected in the LAN. The security deployment allows the directory service to act as a directory proxy to receive and then bypass the authentication requests to the enterprise directory for actual authentication. In order to make such a model work, the source of the directory service has to be examined and verified by the enterprise to gain trust. Once the directory service passes the trustworthiness verification, the service is the only untrustworthy factor in the authentication model, that is, the proposed VFS service is untrustworthy.

In order to gain access to the service, the client has to verify its credential (e.g., username and password) with the enterprise

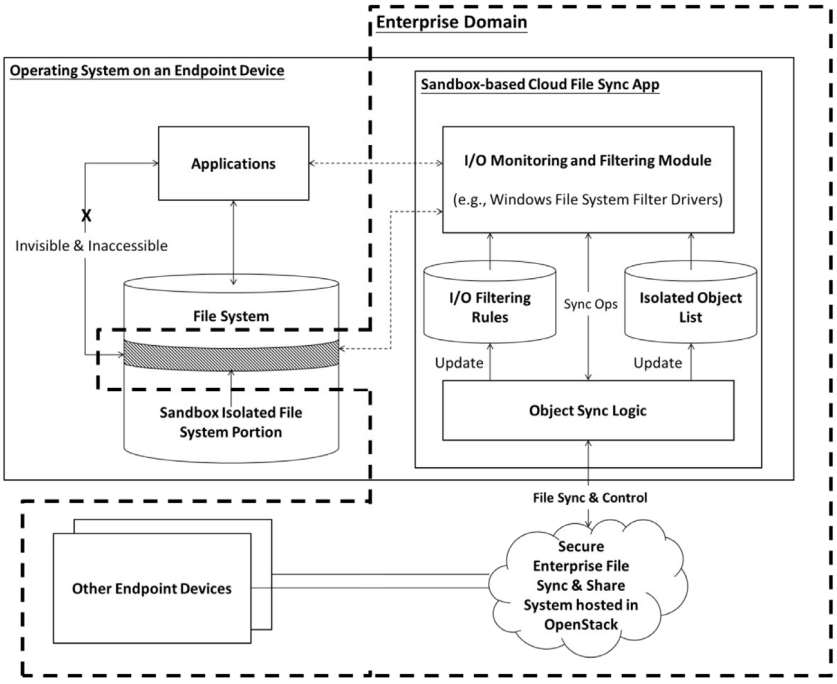


Fig. 7. Sandbox-based cloud file synchronization.

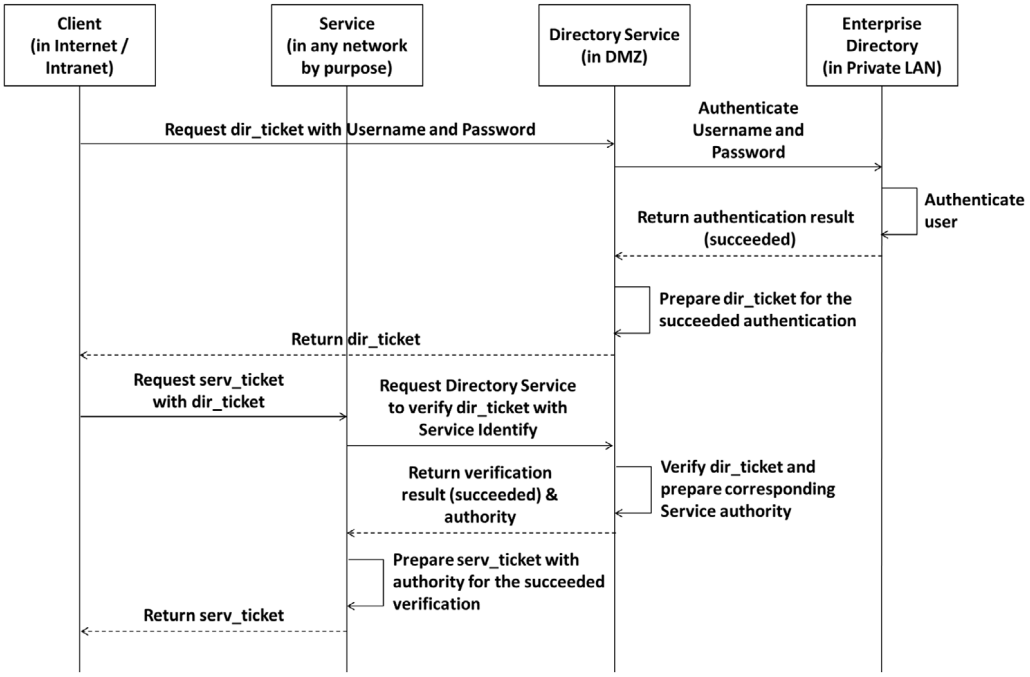


Fig. 8. Sequence diagram of out-of-band authentication.

directory. However, with the secure deployment as shown in Fig. 8, the client has to authenticate the credential with the directory service. After passing the check, the directory service will prepare a directory ticket (*dir_ticket*) to the client for further exchange of a service ticket (*serv_ticket*) with the service. Then, the client can gain access to the service by the service ticket, and the service will be provided to the client based on the authority information provided by the directory service. To conclude, the proposed out-of-band authentication method has the following advantages:

(1) **Security**: Login information (username and password) only pass through managed trustworthy services (e.g., directory service and enterprise directory). Furthermore, the enterprise directory is not associated with any untrustworthy service (e.g., VFS service), which prevents sensitive information stored in the directory service from being accessed by untrustworthy services.

(2) **Flexibility**: The proposed secure deployment is suitable for private and hybrid cloud services. Besides, proprietary requirements of the services can be implemented in the directory service as an extension to the existing enterprise directory. For instance, in order to support additional authentication protocol such as OpenID, the OpenID extension modules can be implemented in the directory service without affecting the enterprise directory.

(3) **SSO**: The directory service can provide multiple services simultaneously, among which the client could have SSO feature by the proposed out-of-band authentication method.

3.6. NoSQL adoption

NoSQL plays an important role in our security framework. Business clouds must be robust enough and resistant to unauthorized attacks, such as SQL injection, a commonly used technique to hack database-driven websites. NoSQL is less vulnerable to SQL injection and other types of hacking. In order to test its robustness, the performances of a NoSQL database, MongoDB, and MySQL are compared under the penetration testing environments, which will be described in the next section. MongoDB is a document-store database, where data are stored in documents rather than rows and columns. JavaScript object notation (JSON) generally stores data, and JavaScript is the language to query data. The following are the advantages of MongoDB: (1) Developers can modify the schema dynamically without causing downtime. In other words, less time can be spent on preparing data for the database and more effort can be spent on making the data work; (2) MongoDB can provide high scalability and allow thousands of transactions to scale up and down easily.

4. Multilayered security

This section describes the multilayered security approach for the CCAF security framework. Our work on software security engineering [7] has motivated the development of a complete framework for cloud security, which is known as CCAF [26]. The CCAF consists of a systematic process for building security and supporting cloud application development life cycle, right from requirements, design, implementation, and testing. The work described in Section 3 has been successfully integrated as the CCAF multilayered security, which has a three-layer implementation model for cloud security as shown in Fig. 1. This model can be applied to both private and public cloud delivery models. The cloud security layers are:

- Access control and firewall (L1), which supports password protection, access control mechanism, and firewall protection;
- Intrusion detection system (IDS) and intrusion prevention system (IPS) (L2), whose aim is to detect attack, intrusion, and penetration, and also provide up-to-date technologies to prevent attacks, such as denial of services (DoS), antispoofing, port scanning, known vulnerabilities, pattern-based attacks, parameter tampering, cross-site scripting, SQL injection, and cookie poisoning. Identity management ensures that only authorized users can gain access to confidential data.
- Encryption/decryption control layer (L3), which supports encryption and decryption of files and messages, including security controls. This feature monitors the system and provides early warning once the fine-grained entity starts to behave abnormally. It also provides end-to-end continuous assurance, which includes investigation and remediation after an abnormality is detected.

Each layer works with a recursive relationship to pass on security control to each other for extra validity and verification of security techniques that are implemented. This can reduce infections by trojans, viruses, worms, and unsolicited hacking, and deny service attacks. Each layer has its own protection, and is in charge of one or multiple functions in the protection, preventive measurement, and quarantine action presented in Fig. 9. The left-hand side of the figure shows the three layers of CCAF security and functionality. Layer 1 (L1) can prevent unauthorized access to synchronized resources and link resources, spoofing, DoS, and port scanning. Layer 2 (L2) can prevent and counter actions for suspicious activities, known vulnerabilities, pattern-based attacks, cross-site scripting, SQL injection, and cookie poisoning. Layer 3 (L3) offers three types of encryption and decryption services: message, file, and full. The right-hand side of the figure explains the relationship between core technologies described in Section 3 and CCAF security. L1 includes sandbox-based synchronization, distinct share link for external share, directory service, and firewall. L2 includes log service, out-of-band authentication with enterprise security, database service, and isolated user storage space for modeling. L3 includes distinct share link for internal share and zero-knowledge system. Once a cloud file access request passed all the three security layers, it reaches the managed OpenStack Swift object storage, and then obtains the requested cloud file. The mechanism of each preventive action of the CCAF security (LHS) matching its corresponding system design (RHS) is illustrated in Fig. 9.

With regard to the technical descriptions in Section 3, the out-of-band authentication belongs to the second layer, because it includes an SSO, directory service, enterprise directory, and OpenID. The authentication on the OpenID double-checks the identity of the users. Upon successful verification, the SSO is activated, and then all data and services can undertake encryption as the third layer. The first layer of security is provided by a login feature to a network and IP-based firewall protocols to ensure security. Access control is achieved by generating a list of users and providing them with different types of user privilege according to their roles. With regard to the system design, our multilayered approach consists of the following three items, and the full connections mapping between each CCAF security layer and core technologies used by the system design can further refer to Table 1:

- Layer 1: Password protection, network, and IP-based firewall and access control
- Layer 2: Out-of-band authentication and OpenID serving for identity management
- Layer 3: Encryption and decryption to ensure that only authenticated files are archived through encryption. Gaining access to any file will require the algorithms presented in Figs. 5 and 6 to be followed.

Table 1 shows the detailed maps between the CCAF security layers and their corresponding core technologies described in Section 3. L1 includes access control, password protection, and firewall to minimize data leakage, unauthorized access, insecure deployment for internal users, and careless sharing of resources with external users. L2 includes detection and prevention of intrusion, and then supports management to detect suspicious activities, report vulnerabilities, strengthen identity management, adopt NoSQL databases, and isolate the user storage modeling. L3 offers file and message encryption and decryption to ensure that all data and passwords are safe and well protected.

4.1. Penetration testing

Penetration testing is commonly used in ethical hacking to validate the robustness of the proposed security prototype and test any vulnerability. The weakest points and vulnerabilities can be detected by the end of the penetrating testing. Several tools and techniques, such as Metasploits, were used to improve penetration testing, and upon successful penetration, known 2012 viruses and trojans were used to test the robustness of the multilayered security. All tests were conducted in the VMs. Comparisons with single-layered McAfee antivirus software were performed to detect any performance issue, such as the number of viruses and trojans blocked and the percentage of blockage. The following data were recorded during the experiment:

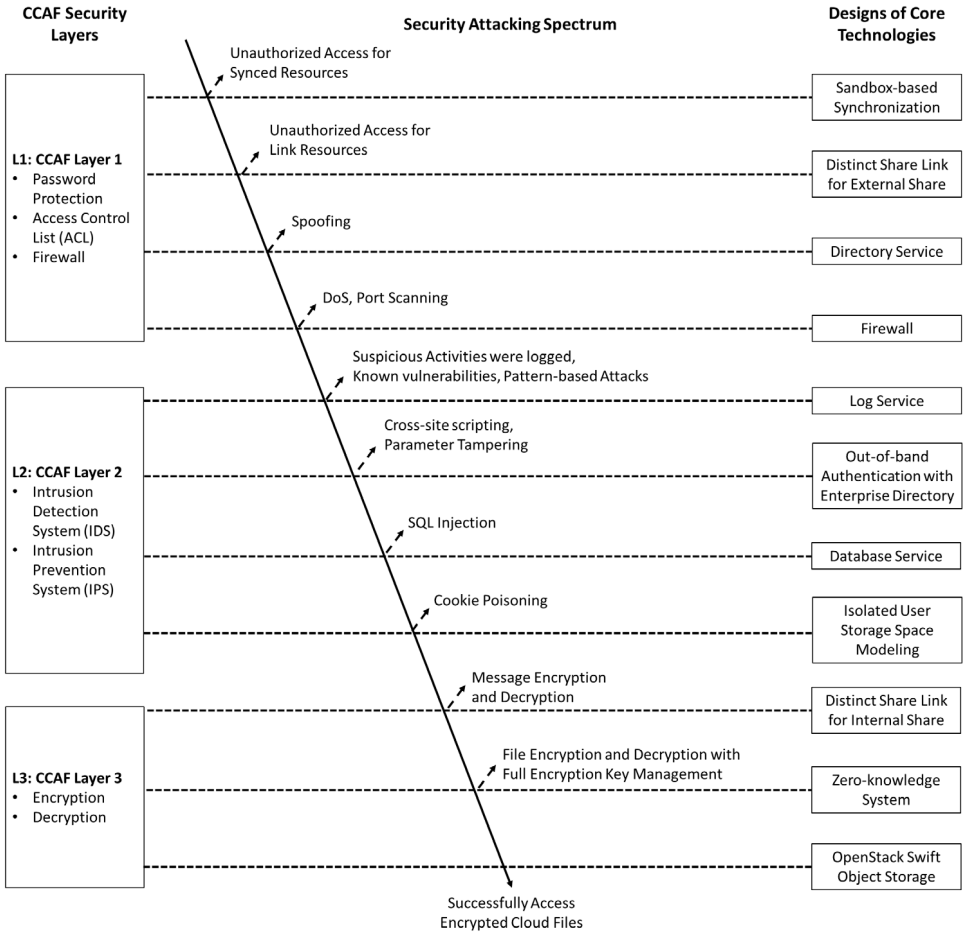


Fig. 9. Three-layered CCAF security.

Table 1
Mapping between CCAF security layers and corresponding core technologies.

Multilayered CCAF		Core technology	
Layer	Capability	Component/mechanism	Targeting security use cases
L1	Access control	Sandbox-based Synchronization	Prevent data leakage to unauthorized domain by limiting file system access at endpoint devices.
L1	Password	Distinct share link for external share	Shared resources are only available to external users who passed password authentication.
L1	Access control	Directory service	Only authorized domains can be applied to shared resources.
L1	Password		Antispoofing by only allowing registered user to login into system.
L1	Access control		Only individual internal services grant own authorized domains.
L1	Firewall	Firewall	Support secure deployment to defend external direct network attacks (such as DoS and port scanning) for internal services.
L2	Intrusion detection	Log service	As a part of intrusion detection system, log service provides audit trail capability to log every activity and event occurred in the system for further investigation. It is useful to detect suspicious activities, such as known vulnerabilities and/or conducting pattern-based attacks.
L2	Identity management	Out-of-band authentication	Incorporate with directory service and enterprise directory to support the identity management capability, particularly the single sign-on service with all internal services. The cross-site scripting and parameter tamping were ended here.
L2	Intrusion prevention	Database service	The adoption of the NoSQL database (MongoDB) provides sort of intrusion prevention capability. Results of an SQL injection testing of the MongoDB will be provided and discussed in the following sections.
L2	Intrusion prevention	Isolated user storage space modeling	Once one of the personal user storage spaces was conquered (by such like the Cookie Poisoning), the isolated user storage space modeling still guarantees other users' data privacy, as all user storage spaces were modeled in own physically separated MongoDB data collections.
L3	Message encryption/decryption	Distinct share link for internal share	Messages for accessing internal distinct share link were encrypted by the recipient's public key before sending, which also prevents network packet inspecting from internal Malware on conquered services.
L3	File encryption/decryption	Zero-knowledge system	Cloud files were encrypted by different random encryption keys, which were managed by the zero-knowledge system with no knowledge about decrypting the cloud files. Only users with correct password (L1) and right identity (L2) in the zero-knowledge system are able to decrypt the original cloud files (L3).

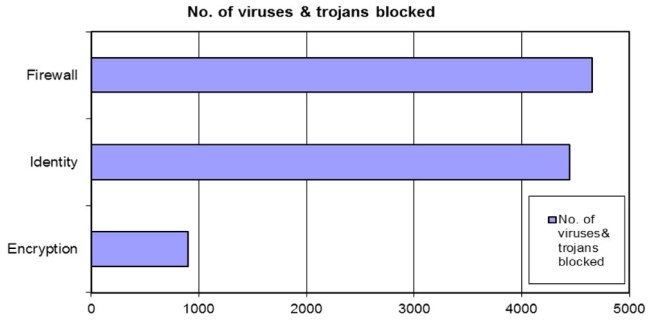


Fig. 10. Number of viruses and trojans detected and blocked.

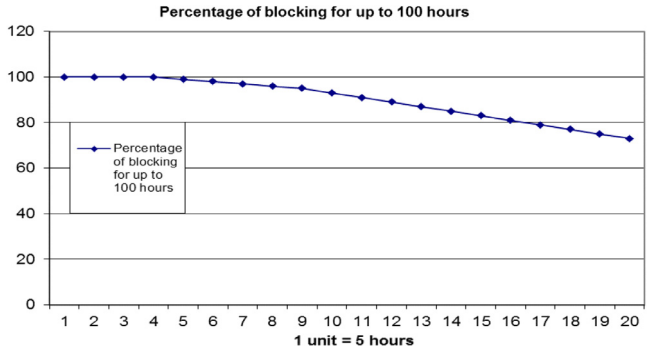


Fig. 11. Percentage of blocking.

- The number of viruses and trojans detected and blocked by each layer.
- The total numbers of viruses and trojans detected and blocked by the system.
- The number of viruses and trojans detected but unable to be blocked and sent to quarantine.
- The number of viruses and trojans that can be destroyed in the quarantine.
- The number of viruses and trojans that cannot be destroyed in the quarantine.

Penetration testing was conducted for multilayered CCAF security. All the trojans and viruses will need to pass the aforementioned three layers. Each layer can detect and filter suspected malicious files and then move to the quarantine area awaiting the next action. With the consolidated defense, all the layers should detect and trap as many viruses and trojans as possible. The purpose of the penetration tests is to identify whether that is the case and the number of true vulnerabilities that the multilayered CCAF security can detect, which are important for business clouds. Two types of experiments were conducted. The first experiment was focused on penetration tests involving injection of 10,000 viruses and trojans in a single attempt. The second one was focused on continuous penetration test, such as injecting 10,000 similar viruses and trojans every 5 h to test the robustness of the CCAF solution.

Fig. 10 shows the number of viruses and trojans detected and blocked by the multilayered CCAF security. A total of 4650 viruses and trojans have been detected and blocked by the firewall, and 4444 have been detected and blocked by identity management and intrusion prevention systems. Finally, 901 viruses and trojans were detected and filtered by encryption, the last step to identify which file in disguise can be detected. The remaining five viruses and trojans were sent to quarantine that they could not be removed or destroyed.

Results are presented in percentages, rather than the number of viruses and trojans blocked. The 10,000 similar viruses and trojans are injected every 5 h to test the mechanism of coping of the data center with the vulnerabilities. Fig. 11 shows that the percentage

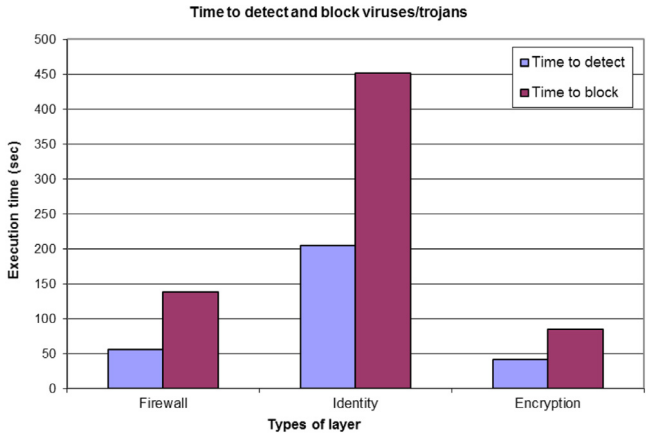


Fig. 12. Execution time to detect and block viruses and trojans.

of viruses and trojans blocked decreased from 100% to 73% in 100 h. This indicates that CCAF multilayered security can withstand regular attacks for up to 5 h with ⩾99% blocking and 10 h with ⩾90% blocking. Performances degraded after 10 h of continuous attack.

The execution times for detecting and blocking viruses and trojans at each layer were recorded. This information is important for the organizations that adopt cloud computing, as the organizations' systems and services will be prone to damage if the vulnerabilities are not detected and blocked early. Fig. 12 shows the execution time to detect and block viruses and trojans in each layer. The firewall layer is the quickest to detect and block vulnerabilities, whereas the identity layer is the slowest, which took >200 s to detect and 451 s to block viruses and trojans. This is because a verification process was required in the identity layer to ensure that the suspected files are viruses and trojans. As the name implies, identity layer involves a verification process to ensure that all files are safe and not in the suspected list. The detected trojans and viruses were blocked and sent to quarantine area for isolation. The reason for the lower execution time of the encryption layer was only the remaining 901 viruses and trojans reach the third layer.

As discussed earlier, 4650, 4444, and 901 viruses and trojans were detected and blocked by the firewall, identity management, and encryption, respectively. The number of detects/blocks per second has to be investigated, which can be calculated by dividing the total execution time in each layer by the number of trojans and viruses detected and blocked (Fig. 13). The number of blocks/detects is the lowest for the firewall due to its design and emphasis in security, whereas it is almost equal for identity and encryption. However, reasons behind this are different. The encryption security used by CCAF and OpenStack could identify the malicious files in disguised forms. Key technologies are based on encryption key management as described in Section 3.3. The malicious files sent to the encryption layer can be further shared with a VM/sandbox environment for testing, which can be destroyed after completion of the test.

Antunes and Vieira [27] use four types of tools for penetration testing and explain the use of precision, recall, and F-measure to determine the validity of their results.

Precision is the ratio of accurately detected vulnerabilities to the number of all detected vulnerabilities:

$$precision = \frac{t_p}{t_p + f_p}. \quad (1)$$

Recall is the ratio of true vulnerabilities detected to the number of known vulnerabilities:

$$recall = \frac{t_p}{t_v}, \quad (2)$$

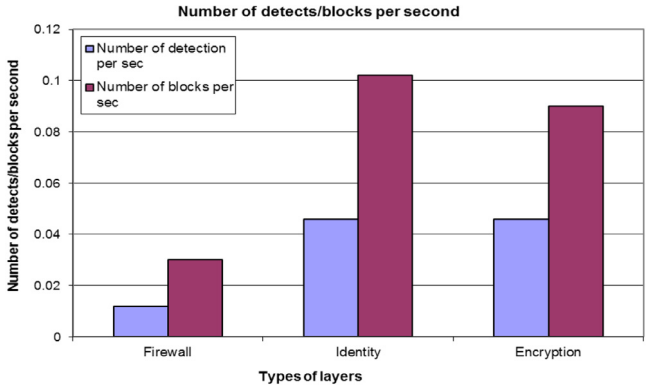


Fig. 13. Number of detects/blocks per second.

Table 2
Comparison of CCAF and other single-layered services.

Services	Precision	Recall	*F*-measure
CCAF	**1**	**0.995**	**0.9975**
VS1	0.455	0.323	0.378
VS2	0.388	0.241	0.297
VS3	1	0.019	0.037
VS4	0.567	0.241	0.338

where

- True positive (t_p) refer to the number of true vulnerabilities detected.
- False positives (f_p) refer to the number of vulnerabilities detected, but do not exist.
- True vulnerabilities (t_v) refer to the total number of vulnerabilities detected in penetration tests.

F-measure can be presented in terms of precision and recall as follows:

$$F\text{-}Measure = \frac{2 \times precision \times recall}{precision + recall}. \quad (3)$$

Better services can generate a high F-measure value [27]. A service with a precision of 0.6 indicates that it can detect vulnerability with 60%. A recall of 0.7 indicates that 70% of the known vulnerabilities are detected. F-measure is found to be 0.646 using Eq. (3). A combination of precision, recall, and F-measure can determine the quality of the security services. We reproduce the experiments conducted by Antunes and Vieira [27] and then compare the results of CCAF multilayered security with those of VS1, VS2, VS3, and VS4 tools due to similarities with CCAF technologies, except that each security is single-layered. Results in Table 2 show that the CCAF multilayered security can provide a much better service as all true vulnerabilities can be detected with a precision of 1. Because only five out of 10,000 viruses and Trojans are missed, the recall is 0.995, resulting in an F-measure of 0.9975, which are higher than the test results. CCAF multilayered security provides better results than other tools and very high precision, recall, and F-measure values, indicating high robustness and validity of security (Table 2).

4.2. SQL injections

Two types of databases are used for multilayered security: (1) a MySQL Server 5.5 that stores database information for users, customers, items, and text-related information and (2) a NoSQL database, the 64-bit MongoDB 3.0.4, storing the same information. The purpose of this ethical hacking is to test robustness of the two databases. This is useful for business clouds where the enterprise security framework, such as CCAF, should retain vital information and prevent unauthorized access or server outage due to attacks. SQL injections are common in unauthorized attacks and are used to perform ethical hacking. There are two ways to improve the performance of SQL injection. First, queries for SQL injection will also create "infinite" loops, thereby generating the database in an erroneous status. Second, 10,000 SQL statements were dumped every second to ensure that SQL injection prevention function is not rectified on time. Combining both methods brings the system down or offline by SQL injection. In order to perform SQL injection more effectively, steps were followed and codes were modified based on suggestions in Ref. [28]. The following three types of tests were conducted for MySQL:

- Using SQL injection with a standard McAfee 2014 edition protection due to the availability of this product.
- Using SQL injection with generic CCAF multilayered security.
- Using SQL injection with full CCAF multilayered security protection that blocks all ports for MySQL, indicating that each SQL query will require authorization and verification.

Tests were conducted for the first hour, followed by the subsequent 24 h. The results of the first-hour test have to be protected, as the information can be leaked and used to hack other parts of the system [29]. The reason for the subsequent 24-h testing is that if a database has been compromised and no remedy has been taken within a reasonable time frame (e.g., 24 h), it will pose threats to the organizations. The entire testing process is performed for both MySQL and MongoDB under laboratory conditions. A rate of 1000 s per simple response indicates that the database has been compromised.

4.2.1. First-hour tests

Response time of MySQL server was recorded to show the effect of SQL injection. If no SQL injection was performed, the normal response time was **0.20 s** per simple query. A very high response time indicates a successful SQL injection. In order to study the effects of SQL injection, one test was focused on finding the variation of response time in the first hour. It is the most critical period to stop SQL injection hacking and prevent any action that makes services and servers down. It is thus concluded that, if the proposed model could not withstand the most critical first hour, it cannot secure data stored in the databases. Fig. 14 shows MySQL (with McAfee) response time during SQL injection, where 1000 represents infinity. The service could not withstand SQL injection after 16 min, resulting in the downtime and extremely slow response to the MySQL server.

Fig. 15 shows that MySQL (with a generic CCAF) response time increases up to 1 h. However, all response times were maintained $\leqslant$12.8 s. Although CCAF multilayered security can minimize the effects of SQL injections, it does not provide full protection to MySQL server. However, the response time was delayed to an acceptable extent of 12.8 s, rather than 0.2 s per simple query. Fig. 16 shows that MySQL (with a full CCAF protection) response time is constant at 0.2 s. In other words, SQL injection could not hack MySQL, as security was ensured in all its ports. For all unauthorized queries, SQL statements would not be accepted. Hence, full protection could offer better protection against unauthorized access or malicious attack.

The next test was focused on SQL injection on a NoSQL database, such as MongoDB, as it could work in the CCAF core technologies described in Section 3. A setup and SQL injection approach similar to MySQL was adopted for the up-to-date version of MongoDB 3.0.4, with three scenarios to be tested. The first test was focused on a MongoDB with McAfee, the second was focused on a MongoDB with a generic CCAF protection, and the third test was focused on a MongoDB with a full CCAF protection. Before beginning the

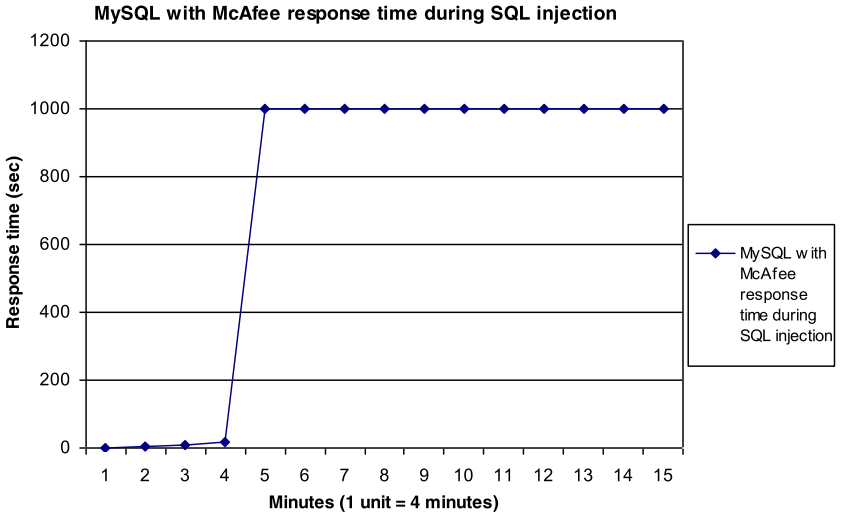


Fig. 14. MySQL with McAfee response time during SQL injection.

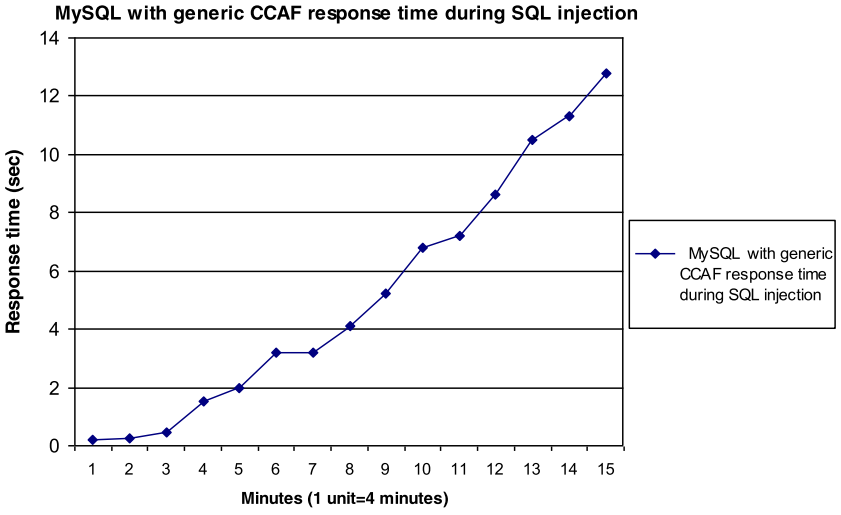


Fig. 15. MySQL with generic CCAF response time during SQL injection.

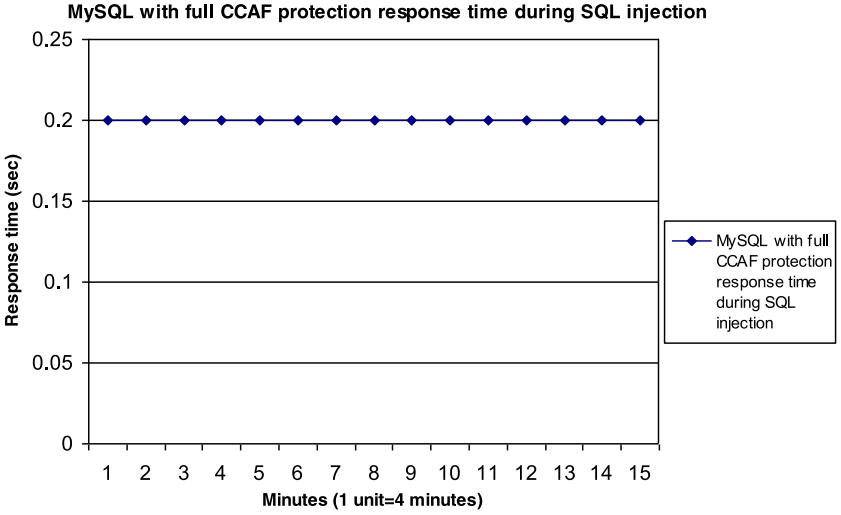


Fig. 16. MySQL with a full CCAF protection response time during SQL injection.

tests, a simple query similar to MySQL SQL injection was used for MongoDB, which could process SQL queries. However, it could take longer, because more time is required to translate it into a query that MongoDB could understand. This was similar to the development of the 64-bit Windows O/S in 2005, which used a portable internal emulator to translate 32-bit applications that can

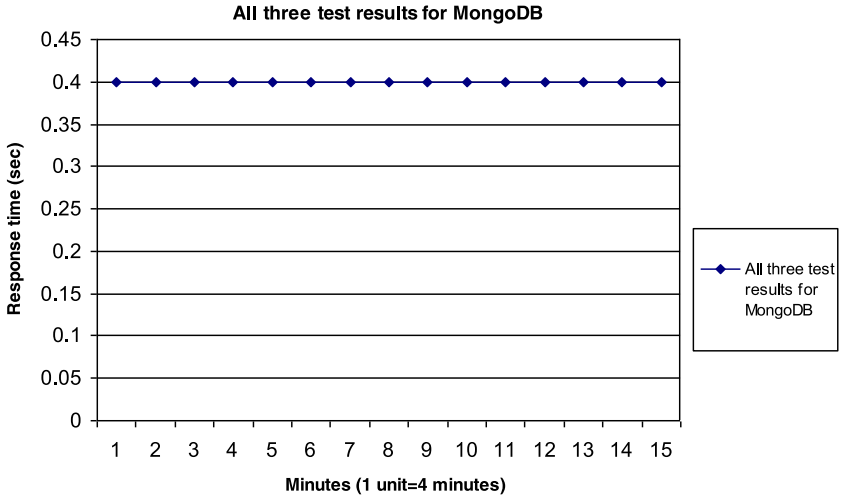


Fig. 17. Three tests of SQL injection for MongoDB.

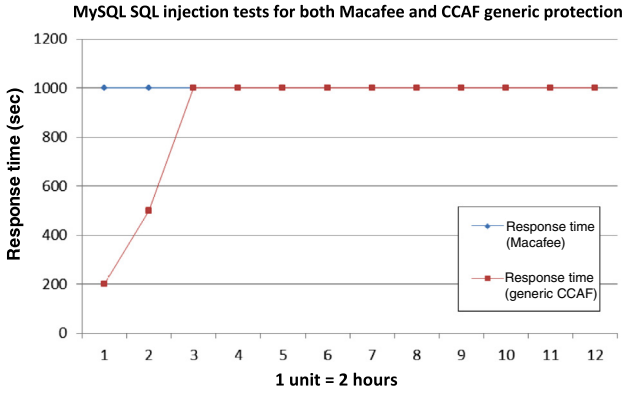


Fig. 18. MySQL SQL injection tests with McAfee and generic CCAF protection test.

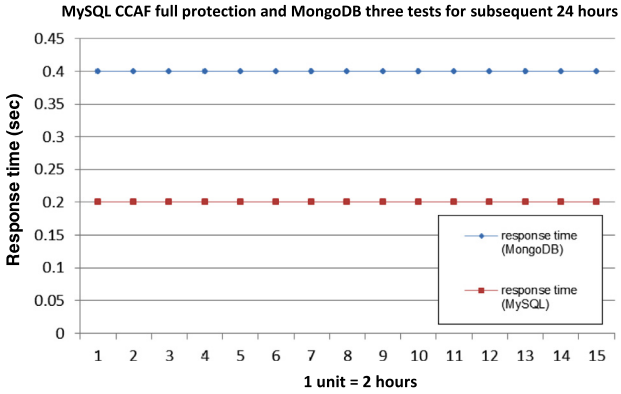


Fig. 19. MySQL and MongoDB SQL injection tests with McAfee and three CCAF protection tests.

be executed on a 64-bit O/S. Each simple query would take **0.40 s** with repeated experiments. The experimental results of the three tests showed that the response time was unchanged, regardless of the security option (Fig. 17).

However, if hackers gain partial access to MongoDB, hacking follows a mechanism as illustrated in Ref. [30]. This indicates that hackers could use social engineering to trick the users believe the real website. Once the passwords are stolen, the hacker could break the MongoDB system console to change security settings. This clearly indicates that MongoDB is vulnerable to security attacks.

4.2.2. Subsequent 24-h tests

This section reports the subsequent 24-h SQL injection tests following the first-hour test. The first pair of test was conducted with SQL injections to MySQL without CCAF protection and MySQL with CCAF protection. Fig. 18 shows MySQL SQL injection tests for both McAfee and CCAF generic protection, similar to the experiments of Figs. 14 and 15 for 24 h. Fig. 18 shows that the response time for CCAF generic protection reaches 1000 s per simple response at the fourth hour, and remained constant thereafter. The response time for SQL injection with McAfee protection was 1000 s per simple response throughout.

Fig. 19 shows MySQL and MongoDB SQL injection tests with McAfee and three CCAF protection tests, similar to the experiments of Figs. 16 and 17 for 24 h. MySQL with CCAF protection can withstand 24 h. Similarly, MongoDB tests with McAfee, generic CCAF protection, and full CCAF protection can withstand 24 h. It shows that any organization adopting business clouds should use MongoDB as the database service. During the migration to NoSQL database, the existing MySQL servers should use full protection, such as CCAF full security, to ensure reliable services.

4.3. Data simulations for CCAF multilayered security

This section describes the mechanism of data management in the multilayered security. Although tests in Sections 4.1 and 4.2 are focused on identifying and blocking malicious files, a large amount of clean and safe data is often transferred through services and servers (in VMs and/or physical machines). Hence, experiments on the use case of transferring and handling a large quantity of clean data are required to be conducted for the following reasons:

- The CCAF multilayered security will not raise false alarm, as it may cause damage to resources, time, and reputation.
- The CCAF multilayered security will identify the clean and safe data and have an excellent performance measurement that allows a large volume of data to be used in the services and servers.

In order to demonstrate that CCAF multilayered security, system design, and experiments on both aspects have been undertaken. The process involved is described as follows: First, the system design to process data was simulated to ensure that all the clean and safe data are checked and processed as quick and efficient as possible. Second, the experiments on using the real data were tested to verify that no false alarm occurs and a short execution time is taken to complete all the checking processes. The CCAF multilayered security was always up-to-date with the

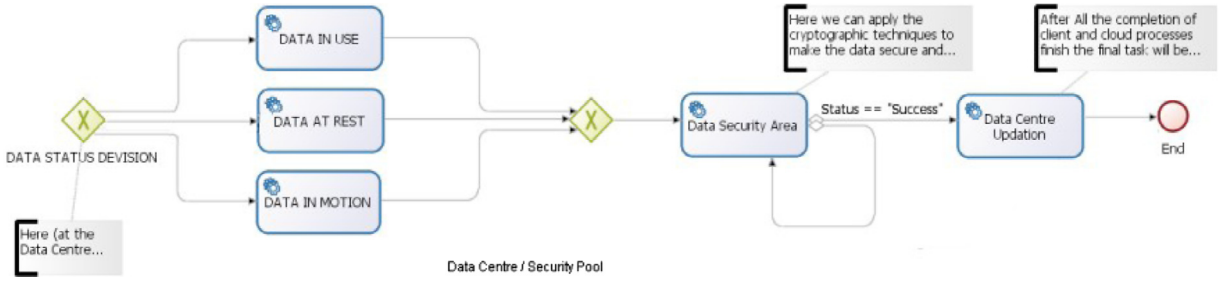


Fig. 20. BPMN model for three types of cloud data security.

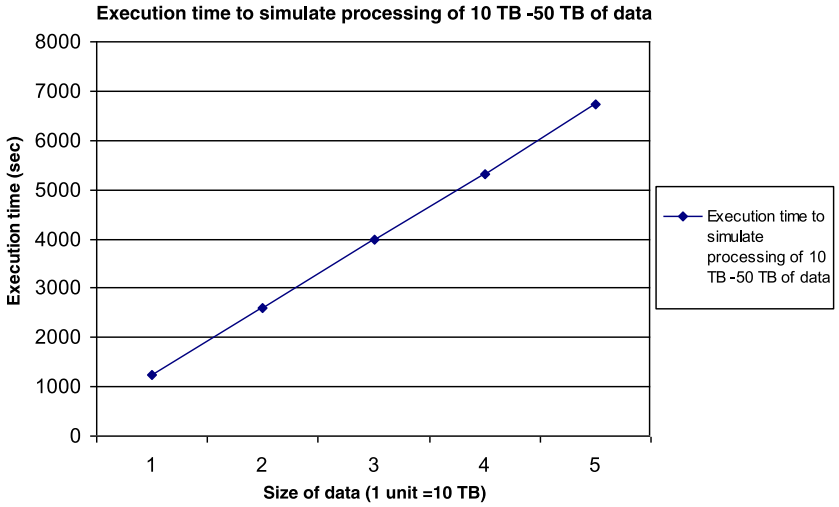


Fig. 21. Execution time to simulate 10–50 TB of clean data.

latest signature to identify the viruses, their sources, behaviors, and infection mechanisms. However, it should take only seconds to identify whether each file is malicious. In order to verify this, real data of between 10 and 50 TB have been used to validate the capabilities of the CCAF service. Data simulation should be performed before using the real data for experiments, so that the organizations that adopt the CCAF service can calculate the time taken to read all the data and verify that they are not infected with viruses and trojans.

In order to facilitate simulation for predicting data management, business process modeling notation (BPMN) was used under the CCAF multilayered security. Fig. 20 shows the BPMN model for analyzing the cloud data security. There are three types of status for data: data in use, data at rest, and data in motion. Data in use indicates that data are used for services or users require them for their work. Data at rest indicates that data are not used for work and are archived in storage. Data in motion indicates that the data are either about to change status from "data at rest" to "data in use" or transferred from one place to another successfully.

All types of data need to satisfy these three conditions at some point. The process starts with a data status decision (diamond symbol) that passes the data to any one of the paths of the cloud storage processes (data at rest, data in use, and data in change/transition). This, in turn, passes the data to a data security pool, which is a separate lane with dedicated security processes (such as data security area and data center update) to study security controls in place before it ends. This simulation (Fig. 20) takes only seconds to analyze terabytes of data. In order to support this further, experiments on simulating 10, 20, 30, 40, and 50 TB of data with BPMN were conducted, and their execution times were recorded.

Fig. 21 shows the execution time to simulate 10–50 TB of clean data. The total time taken was between 1250 and 6742 s, and the relationship between all data points was found to be linear, indicating that the larger the data, the more the execution time.

In order to demonstrate whether the CCAF multilayered security could be subject to false alarm, 10 TB of real data was used for experiments, which included 2 TB of documents and 8 TB of multimedia data, including graphics, images, and videos. For the purpose of testing, all documents were zipped up with a size of 10 GB each, as well as the multimedia. The total numbers of documents and multimedia files used were 50,000 and 10,000, respectively. Each multimedia file was large in size (e.g., 8 TB). All files were checked before the tests by several antivirus tools several times, which reported there was no malicious file. Methods described in Section 4.1, such as the use of precision, recall, and F-measure (Eqs. (1)–(3)), were used to measure the robustness and effectiveness of the results with no false alarm from the CCAF security. Experiments were performed thrice to obtain the average values of the false alarm detected.

The results of all three experiments showed that 59,996 files were reported clean every time. However, four were reported suspected files, but not affected by viruses and trojans. The reason was that those multimedia files were altered by forensic tools, and the altered files behaved differently from ordinary images. These files were considered as warning, rather than false alarm. However, if a strict rule was set, the warning was considered as a partial false alarm, as the service could not fully differentiate between a working file and an infected file, allowing the user's own decision. Hence, two types of results are presented in Table 3. All the values were 1 for tests including warning and 1, 0.9999, and 0.9999 for tests excluding warning. The performance of CCAF was

Table 3
Testing whether false alarm was detected by CCAF multilayered security.

Services	Precision	Recall	*F*-measure
CCAF (including warning)	**1**	**1**	**1**
CCAF (excluding warning)	**1**	**0.9999**	**0.9999**

excellent that it did not respond to false alarm, as supported by the experimental results.

5. Discussion

This section presents six topics for discussion. The first one is the security policy required to deliver the CCAF security framework to the fullest extent. The second is on aligning businesses with governance and information security and discussion about framework approach. The third topic is about integrating the secure EFSS system for the CCAF multilayered security, and the forth one is the general discussion about big data in the cloud. Then, our research contributions will be summed up to demonstrate that the proposed CCAF multilayered security is a valid and working recommendation for business clouds, whereby businesses and institutes should always consider adopting multilayered security approach to ensure that their data and services are guarded with up-to-date protection. Finally, limitations of this study and general approaches from CCAF are discussed.

5.1. CCAF security adoption

The key to successful transformation to cloud computing is the design flexibility of cloud business capable of customizing and adding new security policies. Therefore, it is believed that the adoption of the secure EFSS system – and the security issues that it attempts to address – occupies a critical organizational intersection, between the cost of appropriate countermeasures and the convenience and lower capital cost of operating a business cloud. Within the cloud environment, there has not yet been a standardized approach to security planning. Perhaps, this reflects the shift away from distributed systems to a more logically centralized architecture, which is the characteristic of the cloud. However, there is very little discussion of the role that centralized versus system-specific or issue-specific policies play in the design of secure cloud environments. The security policy lies at the core of a cloud security plan, which should be aiming to reduce risk within budgetary limits, to the extent that management can accept residual risk after countermeasures have been implemented. Marketing security policies that are more or less restrictive could be the basis of competitive advantage. In cloud computing market, there has not yet been any systematic approach to security planning and security policy changes that can be introduced in a systematic manner. According to NIST [11,31], a cloud security plan should encompass:

- Policies, or the set of decisions that the management has made to defend the organization against perceived or measured threats, which can be expressed through standards, guidelines, or procedures.
- Roles and responsibilities, or the matrix that maps who (or what) is responsible for which tasks that need to be undertaken to implement security policy.
- Planning, or the means by which security will be implemented during each stage of a system's life cycle. In the case of cloud storage, this may relate primarily to data life cycle, which imposes a set of stricter standards, particularly relating to privacy.
- Assurance, or determining the extent to which the cloud environment is actually secured.
- Accreditation, or the process of residual risk acceptance by the management.

The multilayered security approach for the CCAF security framework outlined in Section 4 can be mapped from the technical requirements of a set of organizational needs to minimize risk and customization of security policies. At present, this mapping is often ad hoc within cloud environments, relying on the tact knowledge of operational staff to combine the two. Future research will focus on the mechanism of formally specifying these requirements and providing real-time means to achieve assurance against organizational goals. For example, penetration testing should be performed at a level and frequency should commensurate with the value of the data being protected.

5.2. Aligning businesses with governance and information security

A working framework has to align business with governance and information security, so that all the businesses processes and services can minimize risk, reduce errors, improve efficiency and collaboration, and enhance the business opportunities, reputation, and deliveries of services [32]. For example, cloud computing business framework (CCBF) has demonstrated that aligning businesses with corporate, operational, and research activities can result in an increase of efficiency, revenue, and customer satisfaction. Some projects delivered by CCBF demonstrate that the organizations that adopted cloud computing provide services critical to the businesses, such as health informatics, business intelligence, and bioinformatics as a service. Real-life examples for financial modeling and risk simulations to investigate the source of financial crisis in 2009 have been demonstrated [33]. Risk analysis and simulations are parts of enterprise security to help organizations understand what risks are, how to measure, how to analyze, and how to respond to them based on the results. Lessons learned from CCBF have been blended with our current CCAF security to fully integrate multilayered security with governance and information security. All types of risk and security breaches can be informed and alerted in real time to allow swift remedies. A board of governance body can ensure that all the data and services are protected with security updates. Our proposal of the CCAF security is essential to business clouds, because CCAF transforms security concepts into real practice. Examples demonstrated by CCAF can be used by organizations such as University of Southampton and Leeds Beckett University to ensure that all data are safe and protected from real threats.

5.3. Integrating the secure EFSS system for the CCAF multilayered security

The concrete instance and core technologies of CCAF security presented in Section 3 were based on the secure EFSS system, which could be deployed to an on-premise OpenStack cloud environment. One advantage is to offer a dropbox-like service that could back up and transfer the data. Differing from the existing EFSS systems, the designs of the secure EFSS system consider enterprise security concerns, and it has also addressed the EFSS security issues identified in this article. To conclude, key contributions of the proposed integrated security approach are shown as follows:

(1) The firewall friendly secure deployment protects the services and enterprise data.
(2) The isolated personal storage space protects employees' privacies as well as increases system scalability.
(3) The distinct share link supports secure sharing with internal enterprise colleagues and external business partners.

(4) Personal and shared files are encrypted by different keys, which are further protected and managed by built-in key cascading management processes for a secure cloud file sharing environment.
(5) The sandbox-based cloud file synchronization extends the enterprise's domain to employees' endpoint devices to prevent data leaks caused by file synchronization.
(6) Finally, the out-of-band authentication method uses proxy-based authentication to protect the enterprise directory (e.g., AD or LDAP) and prevent unauthorized logging of employees' sensitive data.

5.4. Big data in the cloud

This section describes the relevance of CCAF multilayered security to the special issue: big data in the cloud, a popular topic in the service computing and scalable computing. All security solutions and proposals should allow the organization that adopts cloud computing to offer data security and ensure that all services are active with optimal protection. OpenStack security has been integrated into our CCAF multilayered security. The first layer was focused on the firewall and authentication. The second one was focused on the identity management, and OpenID played an important role in ensuring identity management. The third layer was focused on encryption technologies, as described in Section 3.

A large-scale penetrating test involving 10,000 trojans and viruses was conducted to ensure that the CCAF multilayered security handles a sudden surge of malicious files and blocks them with high accuracy. In addition, it can handle 50 TB of safe and clean data with execution time $<$7000 s. All the experiments had been successfully designed and conducted for big data in the cloud.

5.5. Our research contributions

Initially, a security framework, which can convert theory to practice and a conceptual framework to an architectural framework, is developed. The core technologies are demonstrated and the mechanism of their integrated work is explained. Experiments are conducted to ensure that the proposed design and prototype are robust enough to withstand penetration testing and SQL injection.

Then, a CCAF multilayered security is implemented and blended with core technologies from OpenStack and our existing work. The outcome of the experiments demonstrates that the proposed service can detect and block viruses and trojans much better than existing tools with an F-measure of 0.99. The CCAF multilayered security has not reported any false alarm, which is supported by the results of the experiments.

Finally, the CCAF multilayered security offers research contribution to volume, velocity, and veracity of big data science. It is relevant to volume, as CCAF security can handle 10,000 malicious files. The CCAF security can read 10–50 TB of clean and safe files easily. CCAF security is relevant to velocity, as it can detect and block viruses and trojans in penetration testing and SQL injection ethical hacking. It can respond quickly to unpredicted events resulting from a surge of files and attacks. CCAF security offers veracity, as all experimental results have a high percentage of accuracy, supported by our analysis and results in penetration testing and SQL injection. No attack occurs with the NoSQL database protected by CCAF security.

5.6. Limitations of this research and general approaches from CCAF

The major research limitation is the use of viruses and trojans for penetration testing, which are the 2013 known vulnerabilities. The latest versions are unavailable for testing samples. Collaborators who can provide more up-to-date testing will be sought after for our future work. Other types of penetration testing will be adopted to ensure a better coverage of testing results. The general approaches of CCAF show that the multilayered security is always better and safer than a single security solution for all cloud and big data services.

6. Conclusions and future work

Design and implementation of an integrated security, CCAF multi-layered security, was demonstrated in this article. The motivation and the related literature on the CCAF security were explained, and the core technologies were described, which considered enterprise security concerns and addressed the EFSS security issues. The CCAF security was presented with the integration of the three-layered security: firewall, identity management, and encryption. Experiments were designed to demonstrate CCAF multilayered security as a working framework for business clouds, whose results show that it can detect and block 9995 viruses and trojans during penetration test and could block $>$85% of attacks for 100 h (Fig. 11). Each of the three layers could detect and block viruses and trojans between 0.015 and 0.105 per second. The F-measure was 0.9975, which is higher than that of other services. SQL injection was undertaken for MySQL and MongoDB. Results of the first-hour and 24-h tests showed that a full CCAF multilayered security protection could block and prevent SQL injection for MySQL and MongoDB. All response times were constant at 0.20 and 0.40 s, rather than 1000 s per simple request. Furthermore, the CCAF multilayered security did not detect any false alarm. It took $<$7000 s to read all 50-TB data in the experiments. CCAF security policy could work with real-life examples and also align with businesses to protect assets and data. This study also demonstrated the three major research contributions and explained how it could offer benefits for volume, velocity, and veracity of big data service in the cloud. Our future work is focused on strengthening collaboration with international partners and developing different proofs of concepts, prototypes, services, and research consultancy with our partners.

Acknowledgments

Part of this study was conducted under the "Big Data Technologies and Applications Project (1/4)" of the Institute for Information Industry, which is subsidized by the Ministry of Economic Affairs of the Republic of China. Part of this study was redesigned and experimented with testing and experiments offered by Dr. Victor Chang's private resources conducted in England.

References

[1] X. Zhang, M. Nakae, M.J. Covington, R. Sandhu, Toward a usage-based security framework for collaborative computing systems, ACM Trans. Inf. Syst. Secur. 11 (1) (2008) 3.
[2] H. Takabi, J.B. Joshi, G.J. Ahn, Securecloud: Towards a comprehensive security framework for cloud computing environments, in: 2010 IEEE 34th Annual Conference, Computer Software and Applications Workshops, COMPSACW, July, 2010, pp. 393–398.
[3] T. Zia, A. Zomaya, A security framework for wireless sensor networks, in: Proceedings of the IEEE Sensors Applications Symposium, February, 2006, pp. 49–53.
[4] T. Mather, S. Kumaraswamy, S. Latif, Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance, O'Reilly Media, Inc., 2009.
[5] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, A. Ghalsasi, Cloud computing—The business perspective, Decis. Support Syst. 51 (1) (2011) 176–189.
[6] V. Chang, R.J. Walters, G. Wills, Cloud storage and bioinformatics in a private cloud deployment: Lessons for data intensive research, in: Cloud Computing and Services Science, Springer International Publishing, 2013, pp. 245–264.

[7] M. Ramachandran, V. Chang, Cloud Security proposed and demonstrated by cloud computing adoption framework, in: 2014 Emerging Software as a Service and Analytics Workshop, 2014.
[8] M. Ramachandran, V. Chang, C.S. Li, The improved cloud computing adoption framework to deliver secure services, in: Emerging Software as a Service and Analytics 2015 Workshop, ESaaSA 2015, in Conjunction with CLOSER 2015, Lisbon, Portugal, 20–22 May, 2015.
[9] R.K. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, B.S. Lee, TrustCloud: A framework for accountability and trust in cloud computing, in: 2011 IEEE World Congress on Services, SERVICES, July, 2011, pp. 584–588.
[10] S. Pal, S. Khatua, N. Chaki, S. Sanyal, A new trusted and collaborative agent based approach for ensuring cloud security, 2011. arXiv Preprint arXiv:1108.4100.
[11] NIST, Framework for Improving Critical Infrastructure Cybersecurity, Technical Report, Version 1.0, February, 2014.
[12] S. Subashini, V. Kavitha, A survey on security issues in service delivery models of cloud computing, J. Netw. Comput. Appl. 34 (1) (2011) 1–11.
[13] M. Basso, J. Mann, MarketScope for Enterprise File Synchronization and Sharing, Gartner, 2013.
[14] Y.H. Kuo, Y.L. Jeng, J.N. Chen, A hybrid cloud storage architecture for service operational high availability, in: Proceedings of the 37th IEEE Annual International Computers, Software & Applications Conference, IEEE COMPSAC 2013, July, 2013, pp. 487–492.
[15] J. Sanders, Dropbox and Box Leak Files in Security Through Obscurity Nightmare, TechRepublic, http://www.techrepublic.com/article/dropbox-and-box-leak-files-in-security-through-obscurity-nightmare/, Retrieved on August 15th, 2014.
[16] S.A. Baset, Open source cloud technologies, in: Proceedings of the Third ACM Symposium on Cloud Computing, SoCC, October, 2012.
[17] R. Cattell, Scalable SQL and NoSQL data stores, ACM SIGMOD Rec. 39 (4) (2010) 12–27.
[18] J. Han, E. Haihong, G. Le, J. Du, Survey on NoSQL database, in: Proceedings of the 6th International Conference on Pervasive Computing and Applications, ICPCA 2011, October, 2011, pp. 363–366.
[19] Y.H. Kuo, Y.L. Jeng, Apparatus, Method, and Non-transitory Computer Readable Storage Medium Thereof for Controlling Access of a Resource, U.S. Patent Application (Priority No.: 13/954,885).
[20] Y.H. Kuo, Y.L. Jeng, Cloud Storage Server and Management Method Thereof, U.S. Patent Application (Priority No.: 14/032,440).
[21] D. Dolev, A.C. Yao, On the security of public key protocols, IEEE Trans. Inf. Theory 29 (2) (1983) 198–208.
[22] U.M. Maurer, J.L. Massey, Cascade ciphers: The importance of being first, J. Cryptol. 6 (1) (1993) 55–61.
[23] C. Percival, Stronger key derivation via sequential memory-hard functions, in: The BSD Conference, BSDCan, May, 2009.
[24] Y.H. Kuo, Y.L. Jeng, Secure Synchronization Apparatus, Method, and Non-transitory Computer Readable Storage Medium Thereof, U.S. Patent Application (Priority No.: 14/292,901).
[25] k.Y. Wang, Y.H. Kuo, Y.L. Jeng, Synchronization Apparatus, Method, and Non-transitory Computer Readable Storage Medium, U.S. Patent Application (Priority No.: 14/300,955).
[26] V. Chang, M. Ramachandran, Towards data security with the cloud computing adoption framework, IEEE Trans. Serv. Comput. 8 (6) (2015).
[27] N. Antunes, M. Vieira, Assessing and comparing vulnerability detection tools for web services: Benchmarking approach and examples, IEEE Trans. Serv. Comput. 8 (2) (2015) 269–283.
[28] A. Kieyzun, P.J. Guo, K. Jayaraman, M.D. Ernst, Automatic creation of SQL injection and cross-site scripting attacks, in: IEEE 31st International Conference on Software Engineering, 2009. ICSE 2009, May, 2009, pp. 199–209.
[29] S. McClure, S. Shah, S. Shah, Web Hacking: Attacks and Defense, Addison-Wesley Longman Publishing Co., Inc., 2002.
[30] OWASP article, Testing for NoSQL Injection, Technical Report, accessible on https://www.owasp.org/index.php/Testing_for_NoSQL_injection, July, 2015.
[31] NIST, An introduction to computer security. Publication 800-12, 1996, downloaded from http://csrc.nist.gov/publications/nistpubs/800-12/handbook.pdf.
[32] M.D. Harris, D. Herron, S. Iwanicki, The Business Value of IT: Managing Risks, Optimizing Performance and Measuring Results, CRC Press, 2008.
[33] V. Chang, A Proposed Cloud Computing Business Framework, Nova Science Publisher, ISBN: 9781634820172, 2015, (print).

Dr. Victor Chang is a Senior Lecturer in the School of Computing, Creative Technologies at Leeds Beckett University, UK and a visiting Researcher at the University of Southampton, UK. He is an expert on Cloud Computing and Big Data in both academia and industry with extensive experience in related areas since 1998. He completed a PGCert (Higher Education) and Ph.D. (Computer Science) within four years while working full-time. He has over 70 peer-reviewed published papers. He won £20,000 funding in 2001 and £81,000 funding in 2009. He was involved in part of the £6.5 million project in 2004, part of the £5.6 million project in 2006 and part of a £300,000 project in 2013. He won a 2011 European Identity Award in Cloud Migration. He was selected to present his research in the House of Commons in 2011 and won the best student paper in CLOSER 2012. He has demonstrated Storage as a Service, Health Informatics as a Service, Financial Software as a Service, Education as a Service, Big Data Processing as a Service, Integration as a Service, Security as a Service, Social Network as a Service, Data Visualization as a Service (Weather Science) and Consulting as Service in Cloud Computing and Big Data services in both of his practitioner and academic experience. His proposed frameworks have been adopted by several organizations. He is the founding chair of international workshops in Emerging Software as a Service and Analytics and Enterprise Security. His work has led to creation of two international conferences on Internet of Things and Big Data (IoTBD) and Complex Information Systems (COMPLEXIS). He is a lead Editor-in-Chief (EIC) in International Journal of Organizational and Collective Intelligence and a founding EIC in Open Journal of Big Data. He is the Editor of a highly prestigious journal, Future Generation Computer Systems (FGCS). He is a reviewer of numerous well-known journals and had published three books on Cloud Computing which are available on Amazon website. He is a keynote speaker for CLOSER 2015/WEBIST2015/ICT4AgeingWell 2015 and has received positive support.

Dr. Yen-Hung Kuo was born and raised in Kaohsiung, Taiwan, in 1981. He received his B.S. degree in Information Management from National Kaohsiung First University of Science and Technology (NKFUST), Kaohsiung, Taiwan, in 2004. He earned his Ph.D. degree in the Department of Engineering Science at the National Cheng Kung University (NCKU), Tainan, Taiwan, in 2008. Presently he is a Section Manager, a Principal Engineer of Data Analytics Technology & Applications, Institute for Information Industry, Taiwan. He is serving as a Managing Editor at International Journal on Smart Sensing and Intelligent Systems. Hi is an honor member of the Phi Tau Phi Scholastic Honor Society of R.O.C. He earned the CIE Awards—the Excellent Young Engineers Award, Chinese Institute of Engineers, in 2015. His current research interests include cloud computing, data mining, distributed data management, and software-defined things.

Dr. Muthu Ramachandran is a Principal Lecturer at Leeds Beckett University. He has extensive research coupled with teaching background and experiences on software and systems engineering methods & lifecycle and is an expert in SOA, Cloud and security. He is an author of two books, an edited co-author of a book, and has also widely authored published journal articles, book chapters and conferences materials on various advanced topics in software engineering and education. He is a member of various professional organizations and computer societies. He was also invited speaker on 5th International symposium on SOA Cloud 2012, London.