Dey *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2016) 5:18
DOI 10.1186/s13677-016-0068-6

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

MDA: message digest-based authentication for mobile cloud computing

Saurabh Dey[1*], Srinivas Sampalli[1] and Qiang Ye[2]

Abstract

The emerging area of mobile cloud computing will influence the future of varied applications, such as electronic commerce and health informatics. It is expected to rise in popularity over other models in cloud computing. This is facilitated by its simplicity, accessibility and ease of use. With mobile cloud computing, resource-constrained mobile devices could capitalize on the computation/storage resources of cloud servers via communication networks. Despite the advantage of this innovative computing model, mobile devices in mobile cloud computing are open to more security risks because they often have to access cloud servers through untrusted networks from different locations. Therefore, security is a critical problem to be tackled in mobile cloud computing. One of the most important aspects of mobile cloud computing security is to establish authenticated communication sessions between mobile devices and cloud servers. In this paper, we present a novel authentication scheme, Message Digest-based Authentication (MDA). Technically, MDA strategically incorporates hashing, in addition to traditional user ID and passwords, to achieve mutual authentication. The effectiveness of MDA is validated with Scyther, a widely-used security protocol analyzer. Our experimental results indicate that MDA is capable of withstanding a variety of different security attacks, such as man-in-the-middle, replay attacks, etc.

Keywords: Cloud computing, Security, Mobile devices, Authentication, Hashing

Introduction

The emergence of cloud computing has increased the productivity of a variety of different fields through three service models: SaaS (software as a service), IaaS (infrastructure as a service), and PaaS (platform as a service) [11, 30]. In traditional cloud computing, the following technologies have been used to provide varied services:

- Public Cloud: Public cloud does not require the initial investment and offers a plethora of services to the general mass. However, it does not provide a high level of security for the data, nor does it provide control over the cloud infrastructure [45].
- Private Cloud: Private cloud, as its name indicates, is contrary to public cloud. Private clouds are owned by specific business organizations and are mainly used for internal services. Private cloud can be considered as another form of a traditional server. It provides customized security for data and more control over the cloud infrastructure [18, 45].
- Hybrid Cloud: Hybrid cloud is a combination of both private and public cloud, which is usually used by organizations to improve their resource performance. The organization places its non-important components on the public cloud and controls the computing environment through its private cloud [3].
- Virtualization: Virtualization is the core technology used in cloud computing environment. The cloud server provides the access to a pool of hardware, network and storage resources through virtualization [33]. A virtual platform is created to serve a client with customized configuration and requirements, and by initializing many virtual machines of this kind with limited number of actual physical resources, the cloud server is able to provide its services to multiple concurrent users.
- PaaS (platform as a service): Platform as a service allows product deployment on the cloud, which requires distributed resources [4]. A web-based product might require the access to various resources

*Correspondence: saurabh@cs.dal.ca
[1]Dalhousie University, Halifax, Canada
Full list of author information is available at the end of the article

or infrastructures that are distributed over multiple locations. In order to deploy these products, PaaS provides a platform that has the access to the cloud-based distributed resources.
- IaaS (infrastructure as a service): Using infrastructure as a service, any individual or organization can lease cloud resources to accomplish their tasks [39], such as utilizing storage servers (e.g. renting EMC2 data servers as a storage) provided by a cloud service provider.
- SaaS (software as a service): Software as a service is the most commonly used cloud service, where clients use open or proprietary cloud- based software, such as cloud based anti-virus and word processing software.
- In addition to SaaS, PaaS, and IaaS, cloud service providers offer network-as-a-service (NaaS) [34] by providing the access to virtualized network resources and desktop-as-a-service (DaaS) [15] considering the demand of the availability of desktop resources on mobile devices.

With the advancement of cloud services and mobile technology, it is possible to access and process a large volume of data at any time and from any location. In today's fast-paced environment, this mobility-based complex computing system acts as one of the key components for the growth of research, economy, and society. This combination of cloud infrastructure and mobile technology brings in a new computational model called mobile cloud computing. With mobile cloud computing, resource-constrained mobile devices could capitalize on the computation/storage resources of cloud servers via communication networks. All of the technologies mentioned previously can be used in a mobile cloud computing system.

Despite the advantages of ease of access and 24-7 availability, mobile cloud computing leads to unprecedented security vulnerabilities because data has to be transferred between mobile devices and cloud servers. Therefore, we focus on the security aspect of mobile cloud computing, although many existing studies have attempted to address the behaviour, quality of service, scalability, flexibility, and pricing problem of cloud computing systems [27, 29, 35]. Addressing the security/vulnerability problem in mobile cloud computing is critical to the success of this promising technology.

In this paper, we attempt to improve the security of mobile cloud computing by introducing a novel authentication scheme, Message Digest-based Authentication (MDA). Existing studies indicate that, in general, a mobile cloud computing system uses either user id and password based authentication [36], or USIM (universal subscriber identity module) based authentication [2]. Different from the existing schemes, MDA employs both hashing and the traditional user id/password to arrive at an effective authentication mechanism. MDA ensures that mobile cloud computing is secured from any type of unauthorized access at the beginning of each communication session. This paper is an extension of the previously-published study [38]. As an improvement over the previous study, MDA employs the client registration expiry period and one-way mathematical hashing function to encrypt the user ID in order to make the authentication process more secure and more unpredictable.

The rest of the paper is organized as follows. "Related work" section describes the related work. The details of the proposed authentication scheme, MDA, are included in "MDA: message digest based authentication" section. The evaluation methodology and the in-depth security analysis are presented in "Evaluation methodology and Security analysis" sections respectively. Finally, "Conclusion" section concludes this paper.

Related work

It is important to address the security problems associated with cloud computing in general and mobile cloud computing in particular. Therefore, the related work is categorized into two themes according to the addressed problems. For both of these two themes, the security challenges are highlighted.

Cloud computing security

Popovic et al. describe a number of security issues related to cloud computing [32]. Their study identifies that remote location of resources and virtualization technologies make the cloud computing environment vulnerable to attacks. In cloud computing, all clients access a common resource location, which introduces security threat to the system. In addition, there is an integrity issue in cases of transfer, storage and retrieval. Furthermore, there is a no common standard to ensure data integrity. Different vendors follow different structures for data storage and access, as a result, clients cannot switch vendors easily and remain locked with one vendor. The other mentioned concern is the need for a common standard of encryption and decryption and client's control. Without proper encryption and security, a cloud environment becomes vulnerable, which introduces threat to attack during the transfer and store of sensitive data (identity or location of individual, race, health, union membership, sexual orientation, job performance, biometric, etc.).

Mathisen et al. explain the different policy issues related to cloud computing [30]. If employees of the cloud hosting company are careless and cannot be trusted, then it implies that there is an inherent vulnerability in the service offered. Similar to Popovic et al. this study also mentioned a potential major future concern with vendor lock-in [32]. Since there is no common standard for APIs

(application programming interfaces), storage images and disaster recovery, portability from one vendor to another is an expensive operation. Other than these issues, software used in cloud computing is also an important part, which is vulnerable to external attacks. So by avoiding the use of only open source software in server virtualization, the vulnerability can be reduced.

Yandong et al. focus on different types of clouds viz, public, private and hybrid cloud, and the security issues associated with the public cloud [42]. Public cloud is accessible to a large majority of the general population without much restrictions. This, coupled with the fact that public cloud has plenty of user information, implies that the public cloud is susceptible to attacks.Furthermore, virtualization of the platform introduces security threats. Regulating administrator privileges is required for better authentication, security and privacy. If the virtualization platform is compromised, it implies that a majority of the virtual machines are under attack, which is a potential threat to data security.

To address the above mentioned issues, it is essential to have a standard cloud computing design and usage policy, employee trust, proprietary software for virtualization, and finally physical security of servers. However, data transmitted from a mobile device to a cloud server is still susceptible to several security issues, which will be addressed in the following sub-section.

Mobile cloud computing security

Increased use of mobile devices has meant that extensive research is being performed in the area of security in mobile cloud computing, particularly on authentication. Apart from policy and physical security, there are numerous security issues involved with data transmission and authentication, since mobile devices access cloud from different geographical locations, with different types of networks. The main focus of the study is to design a secure authentication scheme, which is a major component for a secure data transmission process.

Hong et al. describe the mobile cloud security issues related to malicious programs that are injected in a mobile cloud virtualization environment [21]. Virtual mobile instances running on cloud infrastructure are accessed using mobile devices to access resources such as, CPU, memory etc. The research involves usage of machine learning algorithm for detecting abnormal behaviour in the virtual mobile instances. This research is useful to secure the virtualization environment from malware. However, this research does not address issues related to authentication or integrity.

Jana et al. describe the various types of security threats that can appear in a mobile cloud computing environment [23]. Accessing enterprise data through mobile devices brings in privacy and security issues, e.g. locationservices of a mobile device help an adversary to predict the user's behaviour and movement. They focus on various threats related to AAA (Authorization, Authentication, and Accounting), such as malicious insiders, sniffing, spoofing, identity theft etc. As a control measure they suggest digital signature as used in SAML (Security Assertion Markup Language), security gateways, SSL, encryption, hashing. With the help of fishbone analysis they present, security and privacy threats associated with a mobile cloud computing environment.

Yoo et al. highlight about a cellular automata (CA) based lightweight scheme, which is used for multi-user authentication in the cloud environment [44]. The authentication is performed using a one-time password (OTP). For a sender, A, the authentication server sends a seed value ($SeedA$) and a CA rule. The CA is encrypted with the public key of A. Though this approach is OTP based and secure, $SeedA$ is sent from the authentication server to user A without encryption.

Revar et al. describe the usefulness of single sign-on in a mobile cloud computing environment [36]. Single sign-on (SSO) is used on the top layer of cloud. They verified the single sign-on authentication scheme on a Ubuntu server. In SSO, a single user name and password combination is used to access multiple cloud applications. This scheme does not address exactly how a mobile device or any cloud compatible device will access the cloud using SSO.

Ahmad et al. propose a security framework that is based on the subscriber identity module [2]. The authentication is done at boot-time, and is based on USIM response to a random challenge issued by the cloud authentication service. The cloud authentication service has a database to store a list of services, mapping of users and their related services, mapping of IMSIs (International Mobile Subscriber Identity). This scheme works only with mobile devices that support USIM, which is a drawback in case cloud users want to switch to mobile devices that do not support USIM, such as tablets or laptop computers.

MDA: message digest based authentication

In a mobile cloud computing environment, a mobile device has to be registered with a cloud server as a prerequisite process prior to avail any kinds of cloud services. The data transmission between the mobile device and the cloud server must be performed once the mobile device authenticates the cloud server and vice-versa. A strong authentication scheme ensures secure communication between two legitimate parties even if the communication channel experiences potential vulnerability. Since the mobile device lacks of computational capability, it is not suitable to employ complex operations in the mobile device for authentication process. In a mobile cloud computing environment, if a mobile device is registered with a particular cloud service provider, both mobile device

and cloud server must authenticate each other in a uniform way in order to secure the communication with a single authentication scheme, which allows a mobile user to access the cloud server from different locations using different networks and different types of mobile devices. A single and secure authentication scheme will help in preventing third party from posing as a legitimate mobile device or as a legitimate cloud service provider. Therefore, we focus on the following research question (*RQ*).

RQ: Will the use of mathematical hashing, in addition to simple user ID and password authentication improve the security between the mobile device and the cloud server?

To address this research question, we hypothesize that:

H_a: In mobile cloud computing environment, the proposed authentication scheme introduces hashing based security, which reduces the vulnerability of the system to attacks. To determine the vulnerability of the system, we compute the vulnerability score, S_v, which is a measure of the number of attacks a scheme can prevent. The value of S_v lies between 0.0 and 1.0:

$$H_a : 0.0 <= S_v <= 1.0 \quad (1)$$

A low score of S_v indicates that a scheme offers more security. The details of S_v calculation are discussed in "Evaluation methodology" section.

Ahmad et al. mentioned that IMSI can be used as a mode of authentication [2]. However, not every mobile device has the built-in IMSI chip. For example, many iPads and laptops are not equipped with IMSI chips. Therefore, an authentication scheme based on IMSI is not appropriate for mobile cloud computing. Different than IMSI-based approaches, the proposed scheme, MDA, does not require any additional hardware infrastructure. Thus, MDA is applicable to a variety of different mobile devices, including those that do not have IMSI chips. In addition, with MDA, even if the mobile device is stolen, the authentication information of the user can remain to be safe. Furthermore, when users change their registered mobile devices, they can still access the cloud using other mobile devices after a few encrypted files (such as hashed user and cloud certificates, and/or policies) are ported. MDA is composed of three phases: Registration, Authentication and Update. The details of these phases are presented as follows.

Registration phase

The registration process of a mobile device or mobile user to a cloud server is a one time process wherein the user ID and the password are setup and some encrypted files are exchanged. Figure 1 presents an overview of the registration process. Technically, registration (i.e. setting up an account) involves the exchange of user id, password and other unique information, such as credit card for accessing pay-per-use cloud services. We have considered a standard mobile device and cloud server registration process that can be accomplished using any standard protocol, such as SSH (secure shell). Upon setting up the mobile user account with cloud server, the cloud server performs a series of operations. Algorithm 1 shows the detailed registration process adopted in our research.

Algorithm 1 Registration: mobile_with_cloud

Require:
isAlive (*mobile*, *cloud*)
hasNetworkAccess (*mobile*)
Ensure:
Role_Mobile
const *userID*
var *password*
uid = **hash**(*userID*)
pwd = **hash**(*password*)
$cloud \leftarrow$ **send**($uid \| pwd$)
$T_k = uid \oplus pwd$

Role_Cloud
recv($uid \| pwd$)
$T_k = uid \oplus pwd$
EXP = Registration expiry period for the client
$\#CF \leftarrow$ **pointer**{**store**(uid, pwd, T_k)}
MD_{user} = **hash**(usage policy, user access level, user certificate)
MD_{cloud} = **hash**(user add policy, cloud resource restriction, cloud certificate)
MD = **hash**($MD_{cloud} \| MD_{user}$)
$Temp = MD_{user} \| MD_{cloud} \| \#CF \| Pk_{pub\_cloud}$
msz = **encrypt**($Temp, T_k$)
$mobile \leftarrow$ **send**(msz)

The two parties involved in the registration process are mobile device and the cloud server. Both the parties must be alive or be in the network to accomplish the registration process. During this phase, the mobile user's user ID and password are created to access the cloud server. Cloud Server stores $hash\{userID\}$, $hash\{password\}$, and user's mobile device information in big table for efficient lookup. It generates two hashed messages or message digests. The first, MD_{user}, which consists of user policy (cloud resource usage policy, and user access level), and User certificate. The second message digest is MD_{cloud}, and it consists of cloud policy (user add policy, cloud resource restriction/accessibility information), and Cloud certificate.

Upon generating both message digests, cloud server creates an encrypted message to transmit these information to the mobile device. The encrypted is message is: $E_{T_k}\{Pk_{pub\_cloud} \| MD_{user} \| MD_{cloud} \| \#CF\}$, where

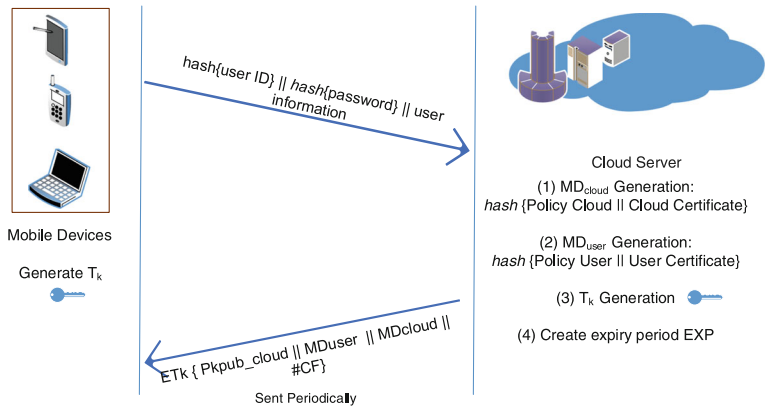

Fig. 1 Registration process of mobile device with cloud server

Pk_{pub_cloud} is the cloud's public key, MD_{user} and MD_{cloud} are the generated message digests, and *#CF* is the column reference, which refers to the cloud authentication database for that particular cloud user information. These information are sent from the cloud server to the mobile device after encrypting with key T_k that is generated in both the mobile device and cloud server by XORing (Exclusive OR) hashed *userID* and hashed *password* (Eq. 2). In addition, the cloud server generates an expiry period for the client registration information, which triggers a periodic update to the authentication parameters and thereby reduces the vulnerability of the system.

$$T_k = hash\{userID\} \oplus hash\{password\} \tag{2}$$

The proposed authentication scheme is applicable once the MD_{user}, MD_{cloud}, and *#CF* are transferred to the mobile device during the registration process.

Authentication phase

In our research, we have the following assumptions:

- The cloud user has two message digests MD_{user}, and MD_{cloud} in the mobile device.
- He/she knows his/her *userID* and *password* to access cloud services.

With these stated assumptions, we can proceed to describe our authentication scheme. At the beginning of each session, the mobile user or mobile device and the cloud server need to authenticate each other in order to start transferring actual data. The authentication process is divided into two steps: cloud authenticating mobile device and mobile device authenticating cloud. The details of these two steps are presented as follows.

Cloud authenticating mobile

The step of cloud authenticating mobile devices can be further split into two sub-steps. In the first sub-step, the following operations are carried out. Specifically, when a mobile device wants to send an authentication request to the cloud server, it generates a key T_k, using hashed *userID* and hashed *password* (Eq. 2). The key T_k, works as the seed for the PRNG (pseudo random number generator) to generate an authentication key, $Auth\_Key_i$. This authentication key $Auth\_Key_i$ is required to encrypt the message digest *MD*, which is generated by hashing MD_{cloud} and MD_{user} (Eq. 3). The $Auth\_Key_i$ is the n^{th} sequence of bits generated by PRNG that is specified by the state identifier *SI*. Key T_k is used to encrypt the state identifier *SI*, and the encrypted message digest $E_{Auth\_Key_i}\{MD\}$. Finally, the encrypted message, $E_{T_k}\{E_{Auth\_Key_i}\{MD\}\|SI\}$ is sent to the cloud server (Fig. 2) along with the column reference *#CF*. Therefore, the message sent from mobile device to cloud server is $\#CF\|E_{T_k}\{E_{Auth\_Key_i}\{MD\}\|SI\}$

$$MD = hash\{MD_{cloud}\|MD_{user}\} \tag{3}$$

Algorithm 2 summarizes the operations involved in the first sub-step. These operations are carried out at the mobile device before the authentication request is transmitted to the cloud server. A flow chart for these operations is included in Fig. 2.

In the second sub-step, the following operations are carried out. Upon receiving the authentication request message, the cloud server performs decryption operations. The cloud server searches the specific hashed *userID* and hashed *password* in the cloud authentication database based on the shared column reference *#CF* that is sent in plain text along with the encrypted message.

Once the hashed *userID* and hashed *password* are found, the cloud server checks the registration validity of the user using *EXP*, stored at the server. If the user registration is not expired, then T_k is generated (Eq. 2) by XOR operation. During this phase, the operations performed at the cloud server are shown in Fig. 3. The generated T_k at the cloud server decrypts the

Algorithm 2 Mobile_to_cloud_auth_req

Require:
isAlive (*mobile*, *cloud*)
hasNetworkAccess (*mobile*)
isRegistered (*mobile*, *cloud*)
uid = **hash**($userID$)
pwd = **hash**($password$)
$cloud \leftarrow$ **send**($uid \| pwd$)
$T_k = uid \oplus pwd$
Ensure:
MD = **hash**($MD_{cloud} \| MD_{user}$)
$Auth\_Key_i \leftarrow T_k$ **PRNG** SI
Enc_1 = **encrypt**($MD, Auth\_Key_i$)
$Temp = Enc_1 \| SI$
Enc_2 = **encrypt**($Temp, T_k$)
$cloud \leftarrow$ **send**($Enc_2 \| \#CF$)

Algorithm 3 Cloud_authenticating_mobile

Require:
isAlive (*mobile*, *cloud*)
hasNetworkAccess (*mobile*)
isRegistered (*mobile*, *cloud*)
Ensure:
$cloud \leftarrow$ **recv**($Enc_2 \| \#CF$)
$SearchEntity \leftarrow (uid \| pwd)$
search($SearchEntity, \#CF$)
if $SearchEntity == found$
  if $EXP == expired$
    triggerUpdatePhase()
  else
    $T_k = uid \oplus pwd$
    $Temp$ = **decrypt**(Enc_2, T_k)
    $Temp = Enc_1 \| SI$
    $Auth\_Key_i \leftarrow T_k$ **PRNG** SI
    MD = **decrypt**($Enc_1, Auth\_Key_i$)
    MD' = **hash**($MD_{cloud} \| MD_{user}$)
    if $MD == MD'$
      integrityCheck($pass$)
      authentication($pass$)
    else
      integrityCheck($fail$)
      authentication($fail$)
    endif
  endif
else
  authentication($fail$)
endif

message $E_{T_k}\{E_{Auth\_Key_i}\{MD\} \| SI\}$ to obtain the state identifier (SI), and encrypted message digest $E_{Auth\_Key_i}\{MD\}$. In addition, the key T_k is used as the seed to the PRNG, and the retrieved SI is used to specify the n^{th} sequence of bits obtained from PRNG as the authentication key, $Auth\_Key_i$. The authentication key, $Auth\_Key_i$ decrypts the encrypted message digests $E_{Auth\_Key_i}\{MD\}$ and obtains MD, which is then matched with the existing message digest stored at the cloud server. Both of the message digests will match only if the user is legitimate.

Algorithm 3 summarizes the operations involved in the second sub-step. These operations are performed at the

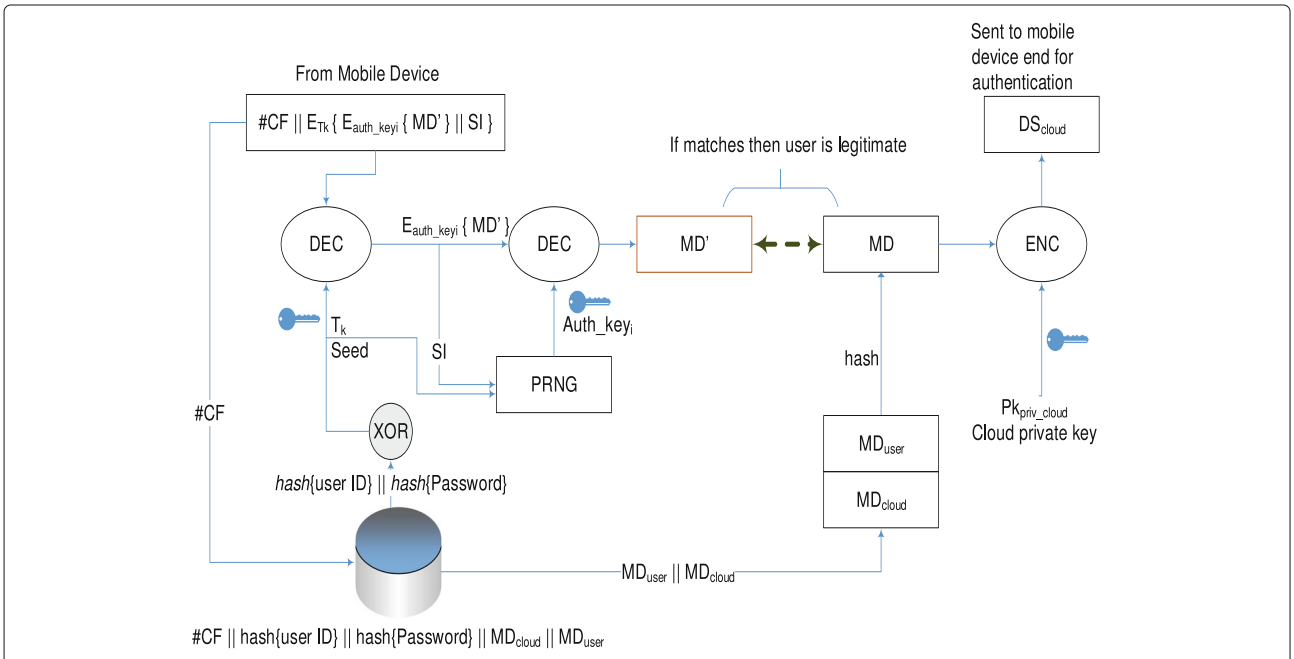

Fig. 2 Mobile device sends encrypted connect request $\#CF \| E_{T_k}\left\{E_{Auth\_Key_i}\{MD\} \| SI\right\}$ to the cloud server, which is encrypted with key T_k [38]

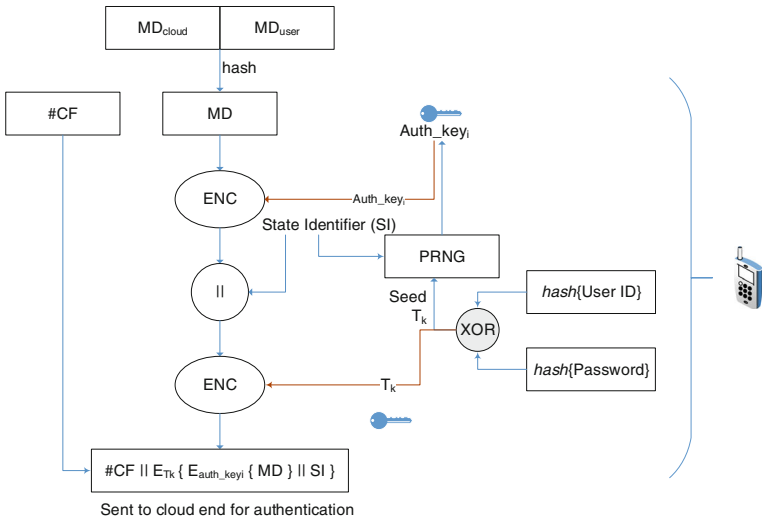

Fig. 3 Cloud server authenticates mobile device when the request is received [38]

cloud server once the authentication request is received. A flow chart for these operations is included in Fig. 3.

Mobile authenticating cloud

Once the mobile device is authenticated, the cloud server sends its digital signature, which consists of MD encrypted with cloud's private key Pk_{priv_cloud}, to the mobile device end. Algorithm 4 summarizes the operations performed at the cloud server during this phase, which includes the creation of digital signature DS and the transfer to mobile devices. A flow chart for these operations is included in Fig. 4.

Algorithm 4 Cloud_to_mobile_auth_req

Require:
isAlive (*mobile*, *cloud*)
hasNetworkAccess (*mobile*)
isRegistered (*mobile*, *cloud*)
Ensure:
MD = **hash**$(MD_{cloud} \| MD_{user})$
DS = **encrypt**(MD, Pk_{priv_cloud})
mobile ← **send**(DS)

After receiving the digital signature DS, the mobile device decrypts it with cloud's public key Pk_{pub_cloud}, stored in the mobile device. If the decrypted MD matches with the message digest MD' stored in the mobile device, then it can be stated that the cloud server is legitimate. Algorithm 5 provides the operations that are performed by the mobile device after receiving cloud's digital signature DS.

On successful execution of the authentication process, both mobile device and cloud server establish a session for data transmission.

Algorithm 5 Mobile_authenticating_cloud

Require:
isAlive (*mobile*, *cloud*)
hasNetworkAccess (*mobile*)
isRegistered (*mobile*, *cloud*)
uid = **hash**$(userID)$
pwd = **hash**$(password)$
$T_k = uid \oplus pwd$

$MD = MD_{user} \| MD_{cloud}$
$msz \leftarrow$ **decrypt**$((MD \| \#CF \| Pk_{pub\_cloud}), T_k)$
$msz = MD \| \#CF \| Pk_{pub\_cloud}$
Ensure:
mobile ← **recv**(DS)
MD' = **decrypt**(DS, Pk_{pub_cloud})
if $MD == MD'$
authentication$(pass)$
integrityCheck$(pass)$
else
authentication$(fail)$
integrityCheck$(fail)$
endif

Update phase

The main purpose of the update phase is to modify the existing authentication entities to increase the degree of unpredictability in the authentication process. This phase is triggered due to reset of the expiry period EXP assigned to each registered mobile client. During the registration phase the cloud server generates an expiry period for each mobile client. This expiry period EXP is checked at the cloud server for each authentication request made by a mobile client. The assigned EXP for a mobile client is basically a real value that decrements over time and if it is reset

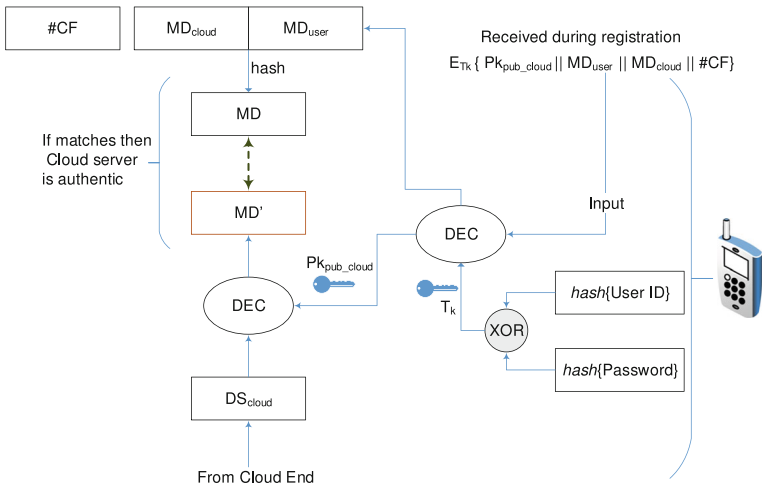

Fig. 4 Mobile device authenticates cloud server using digital signature [38]

or reaches zero then the client is considered as an unregistered client. Apart from assigning an expiry period for a registered client, the cloud server assigns the rate of decrement $rate_{EXP}$ for each expiry period associated with each client. Decrementing the value of EXP by $rate_{EXP}$ is activated by a trigger time tr_{EXP} that enables the decrement process. The cloud server assigns varied expiry periods EXP, rate of decrements $rate_{EXP}$, and trigger time tr_{EXP} for its registered clients, which introduces variability in the expiry period.

If the expiry indicates the registration is expired, then the cloud server sends a re-registration request to the mobile device. The re-registration phase is same as registration phase. However, it verifies the old password and prompts to enter a new password. These information is sent to the cloud server in the similar fashion as the registration phase performs. Upon receiving the re-registration request, the cloud server generates a fresh MD_{user}, and EXP for the client. Since, the password is changed, the new hashed *password* will be different than the old hashed *password* and thereby the generated T_k (Eq. 2) will be different.

Evaluation methodology

We evaluate the proposed scheme by security analysis. By computing the vulnerability score, S_v, we summarize the security offered by the scheme. The proposed scheme is coded using computer simulation environment [9]. We assume that the mobile device is already registered with the cloud server and obtained MD_{cloud}, MD_{user}, $\#CF$, and Pk_{pub_cloud}. In addition, we assume that both the mobile device and the cloud server are running the proposed scheme. During the operation of the proposed scheme, when a registered mobile sends the authentication request $\{E_{T_k}\{E_{Auth\_Key_i}\{MD\}\|SI\}\|\#CF\}$ to the cloud server, or when the cloud server sends authentication response $Pk_{priv\_cloud}\{MD\}$ to the mobile device, it may get compromised by some adversary. The computer simulation provides a way of creating mobile-cloud computing environment, where the initiator is configured as mobile device and the responder is configured as cloud server.

Simulation environment

A research can be performed by performing different types of studies, such as surveys, field study, experiments, computer simulation etc. [9]. We have considered computer simulation to validate our proposed scheme. In one of our simulations, we used a protocol analyzer called *Scyther* [12] as part of the security analayis to launch attacks on the proposed authentication scheme. *Scyther* analyzes a protocol once it is written in *Scyther* coding format. It accepts a minimum of two roles, where one role represents an initiator or sender and the other role represents a receiver. It allows us to define what type of key is used in the protocol, such as symmetric key, or asymmetric key, in addition it allows us to define whether a parameter is constant or variable. Once the protocol is defined, we can configure the attack scenarios. It has many predefined attack scenarios, which are used to test whether a protocol lacks confidentiality and integrity.

Methodology

We compute a vulnerability score (S_v) and we perform a security analysis of the scheme to describe how secure it is. Let us consider N is the number of attacks that are launched on the proposed scheme and $N_{success}$ is the

number of successful attacks that are recorded. Then, the likelihood of successful attacks on the scheme defines its vulnerability score (S_v) (Eq. 4). Lesser vulnerability score indicates that the proposed scheme can prevent more number of attacks. The value of S_v lies between 0.0 and 1.0 (Eq. 1)

$$S_v = \frac{N_{success}}{N} \tag{4}$$

We configured the ted setup and the protocol analyzer *Scyther* for validating the MDA scheme. *Scyther* is configured from the setting option to launch all types of attacks. We analyzed our scheme for 10 RUNs, where both the mobile client and the cloud server have run the scheme to exchange messages. In order to compromise the system, during the execution of each run, *Scyther* tried to launch different types of attacks considering hackers have initial knowledge of the system. We made various security claims, which are validated by running our proposed scheme using *Scyther*. We have recorded the number of attacks to obtain the S_v score.

MDA emulation

We have emulated the operations involved in MDA using Java program and validated it for man-in-the-middle attacks, and replay attacks. The proposed scheme MDA uses public key encryption in the registration process, symmetric encryption in authentication process and a standard hashing algorithm in generating the message digests. MDA does not have any specification for algorithms and therefore, any standard combinations of encryption algorithms and hashing algorithms could be used in the operations of MDA. In order to emulate the operations of MDA, we have used RSA (Rivest, Shamir, and Adelman) as the public key encryption algorithm, SHA1 (Secure Hash Algorithm 1) as the hashing algorithm, and AES (Advanced Encryption Standard) as the symmetric key encryption algorithm. Instances of two user defined java classes (CloudServer and MobileClient) are used to run the operations of MDA.

In addition, we configured the emulated MDA setup with two man-in-the-middle objects. These objects are responsible for modifying the authentication request frames sent from a mobile client to the cloud server, and authentication response frames sent from the cloud server to a mobile device. Upon creation of the mobile device object, we perform the registration process by sending the registration request to the cloud server. The cloud server object uses SHA1 algorithm to generate the client and cloud message digests. RSA Keypair generation is performed at the cloud server and the Pk_{pub_cloud} is sent along with the MD_{user} and MD_{cloud}. Both entities, the mobile device and the cloud server use AES encryption with key T_k during the authentication process. The expiry period *EXP* is set by the cloudserver. For our experiment, we considered only a single client-server model and therefore, we set both tr_{EXP} and $rate_{EXP}$ as zero.

By invoking *run*() method of authentication thread, 1000 authentication request frames are sent from the registered mobile device to the cloud server as shown in Fig. 5. The man-in-the-middle attack object is activated randomly, and it modified the request frames that are sent. Among 1000 frames, 256 frames are modified in transit. The cloud server running MDA scheme has rejected all the modified request frames. Only 744 non modified request frames are processed by the cloud server and 744 authentication response frames are generated and sent back to the mobile device as shown in Fig. 6. MDA is a mutual authentication scheme, therefore, the authentication response frames are authenticated by the mobile device. Further, the second man-in-the-middle object has

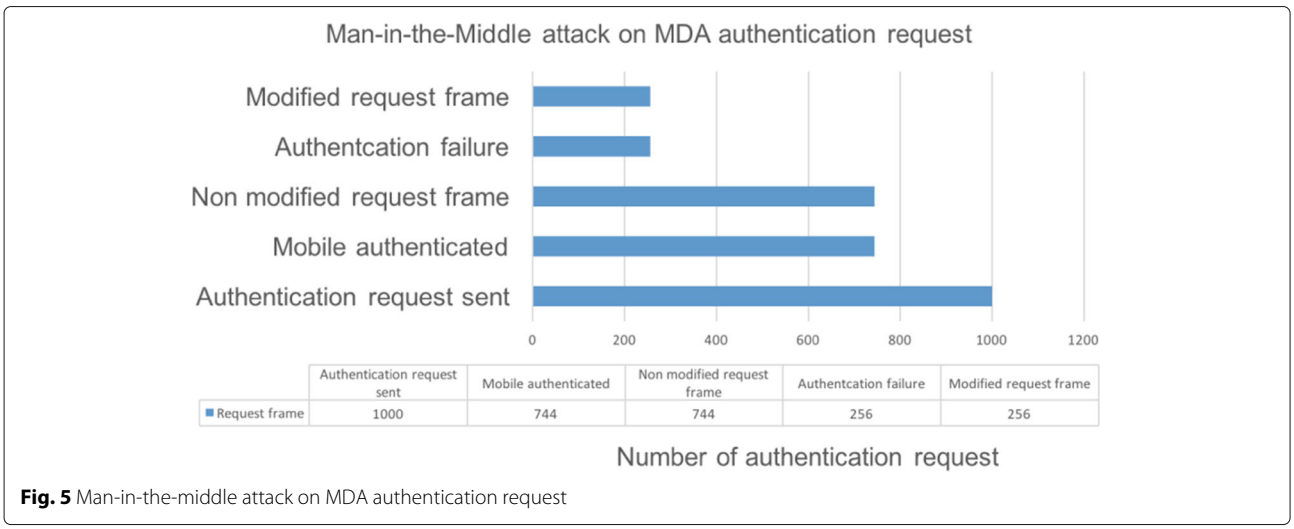

Fig. 5 Man-in-the-middle attack on MDA authentication request

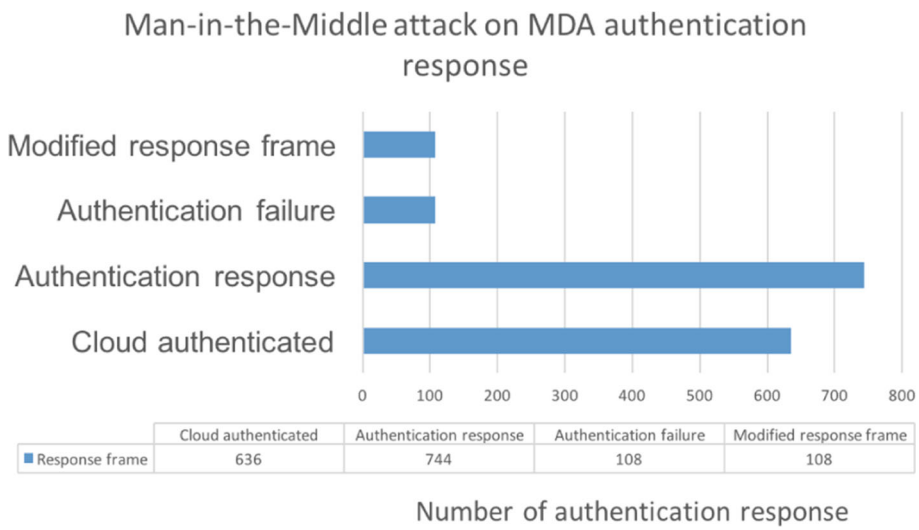

Fig. 6 Man-in-the-middle attack on MDA authentication response

launched random attacks on the response frames. Among 744 response frames, 108 are modified, which are later rejected by the mobile device. Only 636 non modified response frames are accepted. Less than 744 response frames are generated by the cloud server, if we use a non zero positive real value for $rate_{EXP}$, which resets the *EXP* before sending all the 1000 request frames. Although the request frames are non modified and received from registered mobile device, reset of *EXP* forces the cloud server to reject the authentication request. If *EXP* resets, a registration object is initialized to perform the re-registration.

With MDA emulation, we are able to validate the operations of the MDA scheme and exhibit that MDA can withstand man-in-the-middle attacks. However, our emulation has failed to establish that MDA can prevent replay attacks on request or response frames. The scheme does not have any timestamp component, which makes it difficult to identify if a valid request frame is sent from an adversary at later stage. Therefore, replay attacks could be launched on the request frames before *EXP* resets.

Security analysis

The highlight of our scheme is the authentication process. However, the authentication scheme depends on an initial registration phase, which must be reliable and secure in order to keep safe the information required during the authentication phase. We evaluate the vulnerability of the registration phase using *Scyther* and identified that in the absence of security, $hash\{userID\}$, $hash\{password\}$, MD_{user}, MD_{cloud} are falsified and man-in-the-middle attack can be launched. Therefore, in order to establish our approach as a standard authentication scheme in the mobile cloud computing environment, we perform the registration process in a secure manner by employing standard protocol, such as SSH (secure shell). Upon completion of the registration process with SSH, both parties (mobile and cloud server) obtain the necessary parameters for the authentication phase, which is performed using MDA.

To perform the security analysis of the authentication phase, we focus on evaluating the vulnerability of certain parameters such as, $hash\{userID\}$, $hash\{password\}$, MD_{user}, MD_{cloud}, which are used in our authentication scheme. If the vulnerability is high, i.e. these parameters are compromised at any level of communication between the mobile device and the cloud server during the authentication process, then we will fail to establish our hypothesis H_a. In addition, we need to check the aliveness and synchronization of the mobile device and the cloud server during the authentication. We use the security analyzer *Scyther* to check the vulnerability of each of these parameters.

Table 1 summarizes our claims and how secure the scheme is for each claim. In this section, we provide detailed security analysis of the claims that are validated using *Scyther*.

> Claim 1: $Auth\_Key_i$ remains secret throughout the authentication process.
> $Auth\_Key_i$ is used to encrypt and decrypt the message digest *MD* to provide multi layer security . A Mobile device sends the message digest *MD* to the cloud server by encrypting with $Auth\_Key_i$, which is a symmetric key. Both parties, the mobile device and the cloud server can independently generate the

Table 1 Security analysis of the proposed scheme using Scyther during authentication phase

Role	SI.Num.	Claim	Status	Comment
Mobile	1	secret $Auth\_Key_i$	Ok	No Attacks
	2	secret *SI*	Ok	No Attacks
	3	secret *password*	Ok	No Attacks
	4	secret *userID*	Ok	No Attacks
	5	secret MD_{user}	Ok	No Attacks
	6	secret MD_{cloud}	Ok	No Attacks
	7	Alive	Ok	No Attacks
	8	Weakagree	Ok	No Attacks
	9	Niagree	Ok	No Attacks
	10	Nisynch	Ok	No Attacks
Cloud	1	secret $Auth\_Key_i$	Ok	No Attacks
	2	secret *password*	Ok	No Attacks
	3	secret *userID*	Ok	No Attacks
	4	secret *SI*	Ok	No Attacks
	5	Alive	Ok	No Attacks
	6	Weakagree	Ok	No Attacks
	7	Niagree	Ok	No Attacks
	8	Nisynch	Ok	No Attacks

$Auth\_Key_i$ using state identifier *SI*. Since $Auth\_Key_i$ is not exchanged between both the parties, it remains secret.

Claim 2: *State identifier SI is kept secret.*

SI is the state identifier for PRNG used to specify the n^{th} sequence of PRNG as the desired pattern. The mobile device sends *SI* to the cloud server for specifying the n^{th} sequence of PRNG to generate the $Auth\_Key_i$. *SI* is sent from mobile device to cloud server after encrypting with key T_k. The claim at the mobile end that *SI* being secret is validated using *Scyther*. On the other hand, the cloud server does not share or send *SI*, therefore, it is safe at the cloud end.

Claim 3: *password is secret.*

The safety of the *password* depends on the user. Typically no one shares their password, thereby keeping it safe. The password can be a 512 bit string, which is hashed and a copy of the user's *password* is stored at the cloud server during registration process. The mobile device does not send the *password* during authentication process, but it is used at both ends to generate the key T_k. Our *password* safety claim is validated by *Scyther*, but, in reality, it is dependent on the user.

Claim 4: *userID is safe.*

userID is sent to the cloud server only during the registration process, which is required along with the *password* to generate T_k. The *userID* is hashed and a copy is stored at the cloud server during registration process. The mobile device does not send *userID* to the cloud server during authentication process. Therefore, *userID* remains safe. Our *userID* safety claim is validated by *Scyther*, but, in reality, it is dependent on the user.

Claim 5: *The scheme requires MD_{user} to be a secret.*

MD_{user} is the hashed information related to the user, which may contain policy information and unique information about the user. This information is generated at the cloud server end and sent to the mobile device after a successful registration process. The information is hashed and encrypted with $Auth\_key_i$ before transmitting from the user mobile device to the cloud end. *Scyther* validated our claim that MD_{user} is safe.

Claim 6: *The scheme requires MD_{cloud} is secret.*

MD_{cloud} is the hashed information related to the cloud server, which may contain the cloud server certificate or other policy related information. As with MD_{user}, this information is hashed and encrypted with $Auth\_key_i$ before transmitting from mobile device to the cloud server. As part of the authentication process, the combination of MD_{user} and MD_{cloud} is used at the cloud end to verify whether the mobile device is legitimate. In addition, the combination of MD_{user} and MD_{cloud} encrypted with Pk_{priv_cloud} is used as the cloud server digital signature to authenticate the cloud server by the mobile device. The protocol analyzer (*Scyther*) validated the claim that MD_{cloud} is safe.

Claim 7: *Mobile device and the cloud server remains alive during the execution of the protocol.*

The cloud server is said to be *alive* if it has been using the proposed scheme for the initial $(i-1)$ messages exchanged with the mobile device, when the latter sends the i^{th} message. The protocol analyzer (*Scyther*) validates the *aliveness* claim.

Claim 8: *Guarantees Weakagree* [28].

Our proposed scheme guarantees that the mobile device is in *weakagreement* with the cloud server. Therefore, the mobile device is running the proposed scheme with the cloud server; likewise the cloud server is running the scheme with the mobile device. The communication is not affected by adversary during the operation of the proposed scheme. This claim is validated using the protocol analyzer (*Scyther*).

Claim 9: *Assures Niagree between the mobile device and the cloud server* [28].

Niagree or Non-injective agreement ensures that the mobile device and the cloud server agree upon some data exchange during running of the proposed scheme. During the operation of the proposed

scheme, the mobile device can send data to the cloud server safely (and vice-versa). If *Niagree* fails, then we can conclude that there is a man-in-the-middle attack during the operation of the protocol. However, our claim is validated using protocol analyzer (*Scyther*).
Claim 10: *Holds Nisynch during the authentication process* [20].
Nisynch or non-injective synchronization is valid if all actions before the claim are performed as per the proposed scheme description. *Nisynch* ensures that there is no synchronization problem between the mobile device and the cloud server during the authentication process. *Nisynch* is validated using the protocol analyzer, *Scyther*.

Our security analysis results are based on *Scyther* validation, which indicates that MDA supports all of the claims that we have made during the execution of the scheme for multiple runs. The claims are not compromised by any types of attacks during the authentication process. Therefore, the vulnerability score (S_v) obtained is 0.0 , which satisfies the condition of our alternative hypothesis (H_a).

Hashing is a mechanism that makes the proposed scheme secure, and implicitly provides a means for integrity check. Even if any adversary sniffs the message that is sent during the authentication process, it is not possible for the adversary to interpret the message digests, MD_{cloud} or MD_{user} . Furthermore, it is worth noting that MDA is completely based on a technique employing simple *userID*, *password*, and hashing. It does not include any of the system specific properties such as IMSI of mobile phone or MAC (media access control) address of a laptop, which makes the scheme applicable to any type of mobile device without any alteration.

Conclusion

The emergence of mobile cloud computing has transitioned modern computing into a new phase. Mobile devices and cloud resources, the key components of mobile cloud computing, have enabled people to access and utilize the cloud services irrespective of their geographical locations and medium of communications. Numerous fields, such as healthcare, transportation, and financial services, can significantly benefit from mobile cloud computing. Despite the advantages of this innovative computing model, mobile cloud computing could seriously suffer from security/privacy problems. In this paper, we propose a novel authentication scheme for mobile cloud computing, Message Digest-based Authentication (MDA). Technically, MDA is composed of three phases: registration, authentication, and update. With these phases, MDA utilizes hashing, in addition to traditional user id and password based authentication, to ensure confidentiality and integrity during the authentication process. Our analysis results indicate that MDA can survive a variety of different attacks, such as man-in-the-middle, replay attacks, etc.

Acknowledgements
Not applicable.

Authors' contributions
All sections of the paper are written by the author. All authors contributed in the manuscript. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Author details
[1] Dalhousie University, Halifax, Canada. [2] University of Prince Edward Island, Charlottetown, Canada.

Received: 17 May 2016 Accepted: 20 October 2016
Published online: 10 November 2016

References

1. Abolfazli S, Sanaei Z, Shiraz M, Gani A (2012) MOMCC: Market-oriented architecture for Mobile Cloud Computing based on Service Oriented Architecture. In: 2012 1st IEEE International, Conference on Communications in China Workshops (ICCC). pp 8–13. doi:10.1109/ICCCW.2012.6316481
2. Ahmad Z, Mayes KE, Dong S, Markantonakis K (2011) Considerations for mobile authentication in the Cloud. Inf Secur Tech Rep 16(3–4):123–130. ISSN 13634127. doi:10.1016/j.istr.2011.09.009
3. Alizadeh M, Hassan WH (2013) Challenges and opportunities of Mobile Cloud Computing. In: 2013 9th International, Wireless Communications and Mobile Computing Conference (IWCMC). pp 660–666. doi:10.1109/IWCMC.2013.6583636
4. Alrokayan M, Buyya R (2013) A web portal for management of aneka-based multicloud environments. In: Proceedings of the Eleventh Australasian Symposium on Parallel and Distributed Computing - Volume 140. pp 49–56
5. Behl A, Behl K (2012) An analysis of cloud computing security issues. In: World Congress on, Information and Communication Technologies (WICT). pp 109–114. doi:10.1109/WICT.2012.6409059
6. Behl A (2011) Emerging security challenges in cloud computing: An insight to cloud security challenges and their mitigation. In: 2011 World Congress on Information and Communication Technologies. pp 217–222. doi:10.1109/WICT.2011.6141247
7. Benkhelifa E, Fernando DA (2014) On a Real World Implementation of Advanced Authentication Mechanism in a Multi-Tenant Cloud Service Delivery Platform. In: 5th International Conference on Information and Communication Systems (ICICS). pp 1–6
8. Bouayad A, Blilat A, El Houda Mejhed N, El Ghazi M (2012) Cloud computing : security challenges. Colloquium in Information Science and Technology (CIST)
9. Carpendale S, Kerren A, Stasko JT, Fekete J-D, North C (2008) Evaluating Information Visualizations. In: Information Visualization, volume 4950 of Lecture Notes in Computer Science, Springer, Berlin Heidelberg, ISBN 978-3-540-70955-8. pp 19–45. doi:10.1007/978-3-540-70956-5_2
10. Choudhury AJ, Kumar P, Sain M, Lim H, Jae-Lee H (2011) A Strong User Authentication Framework for Cloud Computing. In: 2011 IEEE Asia-Pacific Services, Computing Conference. pp 110–115. doi:10.1109/APSCC.2011.14
11. Chow R, Jakobsson M, Davis UC, Shi E (2010) Authentication in the Clouds: A Framework and its Application to Mobile Users. CCSW10, Chicago
12. Cremers C (2008) The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In: Proceedings of the 20th International Conference on Computer Aided Verification, Princeton
13. Dash SK, Mohapatra S, Pattnaik PK (2010) A Survey on Applications of Wireless Sensor Network Using Cloud Computing. International Journal of Computer Science & Emerging Technologies 1(4):50–55

14. Dinh HT, Lee C, Niyato D, Wang P (2011) A Survey of, Mobile Cloud Computing : Architecture, Applications, and Approaches. Published online in Wiley Online Library. doi:10.1002/wcm.1203/abstract
15. Eaves A, Stockman M (2012) Desktop as a service proof of concept. In: Proceedings of the 13th annual conference on Information technology education - SIGITE '12. p 85. doi:10.1145/2380552.2380577
16. Ficco M, Rak M (2015) Stealthy denial of service strategy in cloud computing. IEEE Trans Cloud Comput 3(1):80–94. ISSN 2168-7161. doi:10.1109/TCC.2014.2325045
17. Guan L, Ke X, Song M, Song J (2011) A Survey of Research on Mobile Cloud Computing. In: 2011 10th IEEE/ACIS International Conference on Computer and Information Science. pp 387–392. doi:10.1109/ICIS.2011.67
18. Joshi JBD, Takabi H, Ahn G (2010) Security and Privacy Challenges in Cloud Computing Environments. Security & Privacy, IEEE 8:24–31
19. Hoang DB, Chen L (2010) Mobile Cloud for Assistive Healthcare (MoCAsH). In: 2010 IEEE Asia-Pacific Services Computing Conference. pp 325–332. doi:10.1109/APSCC.2010.102
20. Hollestelle G (2005) Systematic Analysis of Attacks on Security Protocols
21. Hong JW (2012) Monitoring and detecting abnormal behavior in mobile cloud infrastructure. In: 2012 IEEE Network Operations and Management Symposium. pp 1303–1310. doi:10.1109/NOMS.2012.6212067
22. Huang D, Zhou Z (2011) Secure data processing framework for mobile cloud computing. In: 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). pp 614–618. doi:10.1109/INFCOMW.2011.5928886
23. Jana D, Bandyopadhyay D (2013) Efficient management of security and privacy issues in mobile cloud environment. Annual IEEE India Conference (INDICON), Mumbai, 2013:1–6. doi:10.1109/INDCON.2013.6726077
24. Konstantinou I, Floros E, Koziris N (2012) Public vs Private Cloud Usage Costs: The StratusLab Case. In: Proceedings of the 2Nd, International Workshop on Cloud Computing Platforms. pp 3:1–3:6. doi:10.1145/2168697.2168700
25. Kulkarni G, Gambhir J, Patil T, Dongare A (2012) A security aspects in cloud computing. In: 2012 IEEE International Conference on Computer Science and Automation Engineering. pp 547–550
26. Larosa YT, Chen J-L, Deng D-J, Chao H-C (2011) Mobile cloud computing service based on heterogeneous wireless and mobile P2P networks. In: 2011 7th International, Wireless Communications and Mobile Computing Conference. pp 661–665. doi:10.1109/IWCMC.2011.5982625
27. Lin J-W, Chen C-H, Chang JM (2013) Qos-aware data replication for data-intensive applications in cloud computing systems. IEEE Trans Cloud Comput 1(1):101–115. ISSN 2168-7161 doi:10.1109/TCC.2013.1
28. Lowe G (1997) A hierarchy of authentication specifications. In: Proceedings 10th, Computer Security Foundations Workshop. pp 31–43. doi:10.1109/CSFW.1997.596782
29. Malik SUR, Khan SU, Srinivasan SK (2013) Modeling and analysis of state-of-the-art vm-based cloud management platforms. IEEE Trans Cloud Comput 1(1):1–1. ISSN 2168-7161 doi:10.1109/TCC.2013.3
30. Mathisen E (2011) Security Challenges and Solutions in Cloud Computing, Vol. 5
31. Nkosi MT, Mekuria F (2010) Cloud Computing for Enhanced Mobile Health Applications. In: 2010 IEEE Second International, Conference on Cloud Computing Technology and Science. pp 629–633. doi:10.1109/CloudCom.2010.31
32. Popović K, Hocenski Z (2010) Cloud computing security issues and challenges, Vol. 2010. Opatija, Croatia
33. Qi H, Gani A (2012) Research on mobile cloud computing: Review, trend and perspectives. In: Second International, Conference on Digital Information and Communication Technology and it's Applications (DICTAP). pp 195–202. doi:10.1109/DICTAP.2012.6215350
34. Rangarajan S, Verma M, Kannan A, Sharma A, Schoen I (2012) V2c: A secure vehicle to cloud framework for virtualized and on-demand service provisioning. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics. pp 148–154. doi:10.1145/2345396.2345422
35. Ranjan R, Wang L, Zomaya AY, Georgakopoulos D, Sun X-H, Wang G (2012) Recent advances in autonomic provisioning of big data applications on clouds. IEEE Trans Cloud Comput 3(2):101–104. ISSN 2168-7161 doi:10.1109/TCC.2015.2437231
36. Revar AG, Bhavsar MD (2011) Securing user authentication using single sign-on in Cloud Computing. In: 2011 Nirma University International Conference on Engineering. pp 1–4. doi:10.1109/NUiConE.2011.6153227
37. Sampangi RV, Dey S, Urs SR, Sampalli S (2012) Iamkeys: Independent and adaptive management of keys for security in wireless body area networks. In: Advances in Computer Science and Information Technology, volume 86, Springer, Berlin Heidelberg. pp 482–494. ISBN 978-3-642-27316-2 doi:10.1007/978-3-642-27317-9_49
38. Dey S, Sampalli S, Ye Q (2013) Message Digest as Authentication Entity for Mobile Cloud Computing, 32nd IEEE International Performance Computing and Communications Conference (IPCCC 2013), San Diego
39. Shaikh FB, Haider S (2011) Security Threats in Cloud Computing. In: International Conference for Internet Technology and Secured Transactions (ICITST), (December). pp 11–14
40. Shiraz M, Gani A (2012) Mobile Cloud Computing : Critical Analysis of Application Deployment in Virtual Machines. Int Conf Inf Comput Netw (ICICN 2012) 27(Icicn):11–16
41. Wang S, Dey S (2013) Adaptive Mobile Cloud Computing to Enable Rich Mobile Multimedia Applications. IEEE Trans Multimed 15(4):870–883. ISSN 1520-9210 doi:10.1109/TMM.2013.2240674
42. Yandong Z, Yongsheng Z (2012) Cloud computing and cloud security challenges. In: Information Technology in Medicine … pp 1084–1088
43. Yang L, Cao J, Tang S, Li T, Chan ATS (2012) A Framework for Partitioning and Execution of Data Stream Applications in Mobile Cloud Computing. In: 2012 IEEE Fifth International Conference on Cloud Computing. pp 794–802. doi:10.1109/CLOUD.2012.97
44. Yoo K-Y (2012) A lightweight multi-user authentication scheme based on cellular automata in cloud environment, Vol. 1. doi: 10.1109/CloudNet.2012.6483680
45. Zhang Q, Cheng L, Boutaba R (2010) Cloud computing: state-of-the-art and research challenges. J Internet Serv Appl 1(1):7–18. ISSN 1867-4828 doi:10.1007/s13174-010-0007-6
46. Zhang X, Schiffman J, Gibbs S, Kunjithapatham A, Jeong S (2009) Securing elastic applications on mobile devices for cloud computing. In: Proceedings of the 2009 ACM workshop on, Cloud computing security - CCSW '09. p 127. doi:10.1145/1655008.1655026
47. Zhou F, Cheng F, Wei L, Fang Z (2011) Cloud Service Platform - Hospital Information Exchange (HIX). In: 2011 IEEE 8th International Conference on e-Business Engineering, (2009). pp 380–385. doi:10.1109/ICEBE.2011.35