Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

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

Towards secure mobile cloud computing: A survey

Abdul Nasir Khan [a,*], M.L. Mat Kiah [a], Samee U. Khan [b], Sajjad A. Madani [c]

[a] *Faculty of Computer Science & Information Technology, University of Malaya, Malaysia*
[b] *Department of Electrical and Computer Engineering, North Dakota State University, USA*
[c] *Department of Computer Science, COMSATS Institute of Information Technology, Pakistan*

ARTICLE INFO

Article history:
Received 5 January 2012
Received in revised form
16 July 2012
Accepted 11 August 2012
Available online xxxx

Keywords:
Cloud computing
Mobile cloud computing
Virtualization
Security
Privacy

ABSTRACT

Mobile cloud computing is gaining popularity among mobile users. The ABI Research predicts that the number of mobile cloud computing subscribers is expected to grow from 42.8 million (1.1% of total mobile users) in 2008 to 998 million (19% of total mobile users) in 2014. Despite the hype achieved by mobile cloud computing, the growth of mobile cloud computing subscribers is still below expectations. According to the recent survey conducted by the International Data Corporation, most IT Executives and CEOs are not interested in adopting such services due to the risks associated with security and privacy. The security threats have become a hurdle in the rapid adaptability of the mobile cloud computing paradigm. Significant efforts have been devoted in research organizations and academia to build secure mobile cloud computing environments and infrastructures. In spite of the efforts, there are a number of loopholes and challenges that still exist in the security policies of mobile cloud computing. This literature review: **(a)** highlights the current state of the art work proposed to secure mobile cloud computing infrastructures, **(b)** identifies the potential problems, and **(c)** provides a taxonomy of the state of the art.

1. Introduction

To have an in depth understanding of Mobile Cloud Computing (*MCC*), it is necessary to get a complete grasp on cloud computing. Cloud computing provides a new computing paradigm that delivers *IT* as a service. The objectives of the new computing paradigm are to increase capacity and capabilities at runtime without investing in new infrastructure, licensing new software, and training new recruits. Cloud computing permits customers to utilize cloud services on the fly in pay-as-you-go manner [1,2] through the Internet. The services may be Infrastructure as a Service (*IaaS*), Data storage as a Service (*DaaS*), Communication as a Service (*CaaS*), Security as a Service (*SecaaS*), Hardware as a Service (*HaaS*), Software as a Service (*SaaS*), Business as a Service (*BaaS*), and Platform as a Service (*PaaS*). There are various layered architectures available for cloud computing to provide the aforementioned services as a utility [3–5]. One such cloud computing layered architecture is presented in Fig. 1.

Cloud's backbone layer consists of physical servers and switches. The cloud service provider is responsible to run, manage, and upgrade cloud hardware resources according to the requirements of users. The backbone layer is also responsible to allocate hardware resources to users in an efficient, quick, and smooth way. The supervisor software layer contains the system software to manage the cloud hardware resources. The system software permits application software to run and utilize underlying resources in an efficient way. The various implementations of supervisor software include: (a) operating system, (b) hypervisor, and (c) middleware. The operating system manages the computer hardware resources and provides an interface for interaction of user and application software with hardware resources. The hypervisor is a system software that allows users to remotely create virtual machines on cloud server(s) at runtime. The virtual machine has user defined hardware specifications and a software stack. The virtualization process improves the availability of the user's hosted services even in case of hardware failure. The virtual machine with the entire software stack can be migrated to another server with negligible unavailability of hosted services. Moreover, virtualization is also beneficial for cloud service providers. Because, *IT* research firm "Infotech" estimates that the distributed physical servers without virtualization utilize only 20% of total capabilities. The virtualization process can boost hardware utilization between 60% and 80% [6]. The middleware system software manages the transparent execution and interaction among jobs running on cloud servers. The software infrastructure layer hands over the network resources to upper layers and provides a foundation for a new computing paradigm that delivers *IT* as a service. The software infrastructure layer can be subdivided into three categories: (a) *IaaS*, (b) *DaaS*, and (c) *CaaS*. *IaaS* [7] offers computational

* Corresponding author. Tel.: +60 108124496; fax: +60 379579249.
E-mail addresses: anasir@siswa.um.edu.my (A.N. Khan), misslaiha@um.edu.my (M.L. Mat Kiah), samee.khan@ndsu.edu (S.U. Khan), madani@ciit.net.pk (S.A. Madani).

0167-739X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2012.08.003

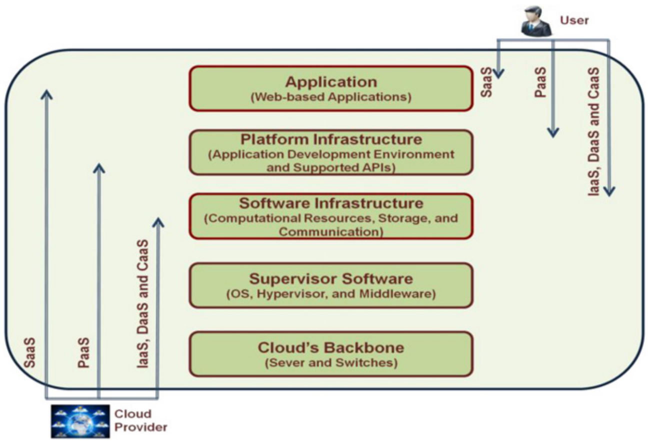

Fig. 1. Layered architecture of cloud computing [6].

resources (e.g. servers) to the end users. For efficient utilization of resources, multiple virtual machines may be created on each server according to users' demand. The examples of *IaaS* include Amazon Elastic Computing Cloud (*EC*2), Enomaly's Elastic Computing Platform (*ECP*) [8], and Resources and Services Virtualization without Barriers (*RESERVOIR*) architecture [9]. *DaaS* allows users to store an enormous amount of data on remote server(s). The users can access the uploaded data immediately from any site having Internet connectivity. Simple Storage Service (*Amazon S*3) [10] and Zenter ZumoDrive [11] are the examples of *DaaS*. *CaaS* provides secure, reliable, and fast communication services for the users. The platform infrastructure layer provides an application development platform and a set of Application Programming Interfaces (*APIs*) for the developers. The developers use *APIs* to program applications that can interact and utilize the full power of the cloud resources. Google App Engine [12] and Salesforce.com's Apex Code [13] are examples of *PaaS*. The topmost application layer allows users to access and use applications installed on a cloud provider's data center through the Internet. Users can easily access such applications through a web client, even with limited processing and storage capabilities. Moreover, a cloud service provider can upgrade the applications without requiring the user to install any update or patch.

Cloud computing with resource constraint mobile devices, ubiquitous wireless infrastructure, mobile web, and location-based services provides a ground for a new computing paradigm called *MCC*. Mobile devices being battery powered, have limited processing power, low storage, less security, unpredictable Internet connectivity, and less energy. The aforementioned limitations of mobile devices are always obstacles for computationally intensive and storage demanding applications on a mobile. To augment the capability, capacity and battery time of the mobile devices, computationally intensive and storage demanding jobs should be moved to cloud [14,15]. On the basis of the above discussion, *MCC* can be defined as:

"A service that allows resource constrained mobile users to adaptively adjust processing and storage capabilities by transparently partitioning and offloading the computationally intensive and storage demanding jobs on traditional cloud resources by providing ubiquitous wireless access".

Careful planning is required before offloading the jobs on a cloud server by considering the network conditions and communication overhead to make offloading beneficial for mobile users [16]. The architecture of the *MCC* is depicted in Fig. 2.

Mobile clients interact with a cloud service provider using native mobile applications or embedded browser applications. Embedded browser applications are developed using standard web development languages (e.g. HTML and JavaScript). Native applications are developed using mobile platform supported programming languages and a set of *APIs* provided by the cloud service provider. Mobile clients are connected with the base transceiver station to access the mobile network services. The mobile client utilizes mobile network services to communicate with cloud through the Internet. There are two types of cloud servers: (a) portal cloud server(s) and (b) back-end cloud server(s). The portal cloud server receives and processes the mobile client requests for using the cloud services. The portal cloud server is also known as the cloud controller in literature. The portal cloud server utilizes the back-end cloud servers for providing different services to mobile clients.

1.1. Security threats and counter measures

There are numerous challenges existing in the field of *MCC*, including data replication, consistency, limited scalability, unreliability, unreliable availability of cloud resources, portability (due to lack in cloud provider standard), trust, security, and privacy [17]. The above-mentioned challenges have become a barrier in the rapid growth of *MCC*'s subscriber. According to survey [18,19], 74% of *IT* Executives and Chief Information Officers are not willing to adopt cloud services due to the risks associated with security and privacy. To attract potential consumers, the cloud service provider has to target all the security issues to provide a completely secure environment. Research organizations and academia have undertaken a massive amount of work to secure a cloud computing environment. There are still some grey areas that need to be addressed, such as the security and privacy of user's data stored on cloud server(s), security threats caused by multiple virtual machines, and intrusion detection. As *MCC* is based on cloud computing, all the security issues are inherited in *MCC* with the extra limitation of resource constraint mobile devices. Due to resource limitation, the security algorithms proposed for the cloud computing environment cannot be directly run on a mobile device. There is a need for a lightweight secure framework that provides security with minimum communication and processing overhead on mobile devices. Fig. 3 shows the different security services that may run on different layers to provide a secure *MCC* environment.

The security and privacy protection services can be achieved with the help of secure cloud application services. In addition to security and privacy, the secure cloud application services provide the user management, key management, encryption on demand, intrusion detection, authentication, and authorization services to mobile users. There is a need for a secure communication channel between cloud and the mobile device. The secure routing protocols can be used to protect the communication channel between the mobile device and cloud. Virtualization improves the utilization of cloud resources but introduces new security issues due to the lack of perfect isolation of virtual machines hosted on a single server. The security issues imposed by virtualization can be tackled to some extent with the help of virtual machine secure monitoring, mirror, and migration. To provide the transparent cloud environment, mobile users must have the facility to audit the security level of the hosted services. The audit can be done with the help of a cloud service monitor. The cloud service monitor examines the security level and flows of the running environment. The security level should meet the user security requirements and the flow of the running environment should be normal. The security verification of uploaded data on cloud can be done using a storage security verification service. The physical security of the datacenter plays a very important role to achieve security and privacy. Physical security deals with the measures taken to avoid unauthorized personnel physically accessing the resources of the cloud service provider. Physical security can be achieved with the help of security guards, video surveillance,

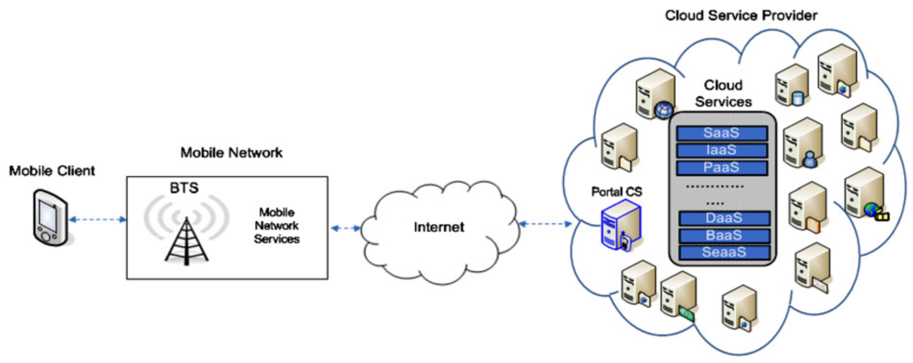

Fig. 2. Mobile cloud computing architecture.

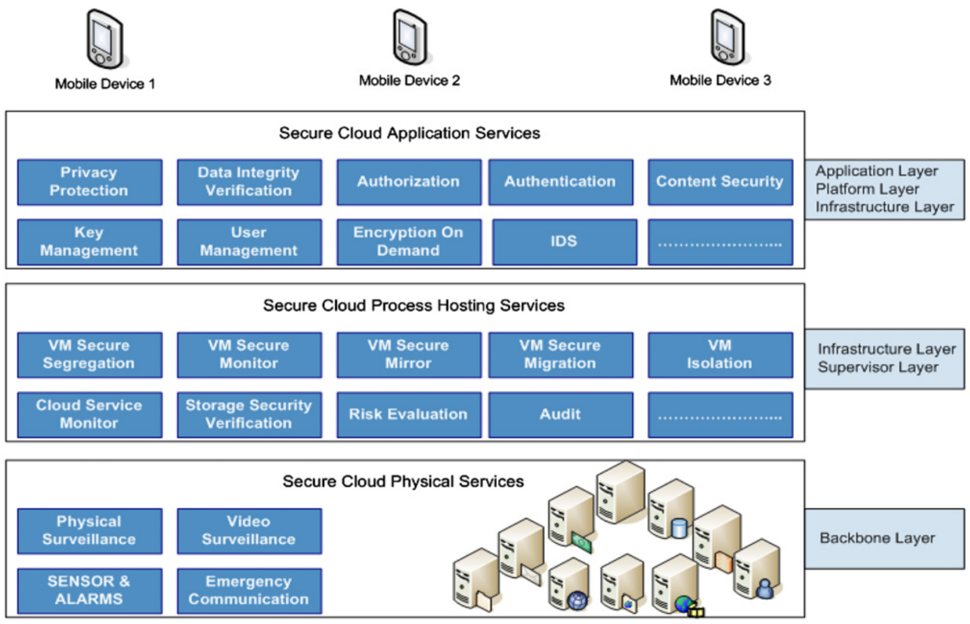

Fig. 3. Security services on different layers.

security lighting, sensors, and alarms. The researchers have done a massive amount of work [20–28] to provide an energy-aware high performance computing environment. However, there is also a need for energy-efficient security frameworks for mobile devices to provide security and privacy services in an *MCC* environment [29].

For the last few years, *MCC* has been an active research area. As *MCC* is in the preliminary stages, limited surveys are available in various domains of *MCC* [30–32]. In [30], the authors present the comparison of different application models for *MCC* and highlight the research challenges on the subject. The authors in [31] have investigated the *MCC* architectures, application offloading, and context-aware services. To the best of our knowledge, there exists only a single survey [32] that discusses security issues as a part of the survey in *MCC*. The major focus of the survey presented in [32] is on mobile communication and computing issues. The mobile communication issues are associated with low bandwidth, service availability, and heterogeneity. The computing issues are linked with computing offloading, security, data access, and context-aware mobile cloud services. This survey is different from [32] in the following aspects: (a) the focus of the survey is completely on security issues in *MCC*, (b) existing lightweight security frameworks are investigated in detail, (c) the survey highlights important parameters and discusses the impact of the parameters on security frameworks, (d) the survey identifies the open research issues regarding security and privacy in *MCC*, and (e) state of the art taxonomy is presented.

The rest of the paper is organized as follows. Section 2 presents the parameters used to evaluate different security frameworks for *MCC*. Section 3 deals with actual survey of different security frameworks that have been presented and published. The positive and negative aspects of security frameworks are illustrated in Section 4. Finally, Section 5 concludes our survey.

2. Evaluation criteria for security frameworks

A number of the security frameworks presented in the survey deal with the security of files/data created and manipulated on a mobile device or cloud servers. The rest of the frameworks cover the security aspects of mobile applications or a mobile application model that uses cloud resources to augment the capability of a mobile device. Therefore, the survey classifies existing security frameworks for *MCC* into two categories: (a) data security frameworks and (b) application security frameworks. The computational requirements, scalability, and assumptions play a very important role for the successful deployment of security frameworks in an *MCC* environment. Moreover, mobile users are more interested in storing files on a cloud server without exposing any information. Security frameworks should also provide privacy and security features to mobile users.

2.1. Evaluation criteria for data security frameworks

The data security frameworks deal with the security of mobile user's files created and manipulation on a mobile device or cloud server. By considering the importance of the aforementioned parameters (Section 2) in *MCC*, the following evaluation parameters have been selected for comparing the presented security frameworks.

2.1.1. Basic theory

The basic theory parameter specifies the basic building blocks of the discussed security frameworks for *MCC*. The basic building blocks may be mathematical or cryptographic principles. The basic theory parameter is included to identify the computational requirements of the discussed security frameworks.

2.1.2. Data protection

A number of security frameworks cover the Protection of Data Created and Manipulated on Device (*ProDCMD*) [29]. The rest of the security frameworks deal with the Protection of Data Created and Manipulated on Cloud (*ProDCMC*) [33]. The data protection parameter identifies the category of discussed security frameworks as *ProDCMD* or *ProDCMC*.

2.1.3. Data integrity

To increase the storage capacity, mobile users upload files on the cloud server and lose physical control of uploaded files. There should be a mechanism to ensure the correctness of users' uploaded files. The correctness of the uploaded file can be verified with the help of integrity verification. The data integrity parameter identifies the consideration of the integrity verification issue in discussed security frameworks.

2.1.4. Scalability

Scalability is the ability of the system to handle a growing amount of users in an elegant manner. The security framework is considered to be highly scalable, if the users' increase can be adaptively handled without degradation in performance or change in physical infrastructure. If the proposed security framework is dependent on some centralized server managed by a third party to provide security features, the scalability of the framework is considered as moderate, otherwise poor.

2.1.5. Assumption

A security framework is considered to be secured, if the underlying assumptions of the framework are weaker. The assumption parameter identifies the components that are assumed to be fully trusted, semi-trusted, or distrusted to provide security features in an *MCC* environment. Semi-trusted means some functions are assumed to be done perfectly but some may be compromised, for example storage may be exposed but computation is properly conducted.

2.1.6. Data access

Data access can be divided into three categories: (a) automated and (b) semi-automated. Data access is considered to be automated, if users share encrypted files located on cloud servers among groups of people and authorized users can access and decrypt files automatically (without the physical involvement of the file's owner). The data access is considered to be semi-automated if the user requires to send some secret information (e.g. password, secret key) through other means (e.g. email, SMS, or call) to access and decrypt the uploaded file.

2.1.7. Authentication

If a mobile user has uploaded the files on cloud server for sharing with multiple users, there should be a mechanism to verify the originator of the file. The authentication mechanism may help to verify the originator of the file. The authentication parameter identifies the consideration of authentication issue in discussed security frameworks.

2.2. Evaluation criteria for application security frameworks

The application security frameworks deal with the security of the mobile application or mobile application model [34–37] that utilizes cloud resources to provide better services to mobile users. The selected evaluation parameters are discussed below:

2.2.1. Application type

Application type parameter is used to identify the mobile application model or type of mobile application whose security aspects are covered in the *MCC* environment.

2.2.2. Security features

Security features parameter identifies the covered security aspect of mobile applications or mobile application models in the *MCC* environment. The security features may be data security, data integrity, identity privacy, location privacy, authentication, secure data access management, risk management, or secure routing.

2.2.3. Assumptions

The framework is considered to be secured, if the underlying assumptions of the frameworks are weaker. The assumptions' parameter identifies components that are assumed to be fully trusted, semi-trusted, or distrusted to provide security features.

2.2.4. Scalability

Security frameworks proposed for mobile applications or the mobile application model are considered to be highly scalable, if the users' increase can be adaptively handled without degradation in performance or change in physical infrastructure. If the proposed security framework is dependent on some centralized server managed by a third party to provide security features, the scalability of the proposed framework is considered moderate, otherwise poor.

3. Survey of existing security frameworks for *MCC*

In this section, we present countermeasure solutions that have been proposed in the scientific journals and conferences pertaining to securing *MCC*. A comparison and critical discussion on the proposed ideas will be detailed in the next section.

3.1. Data security frameworks

The countermeasures for data security frameworks are presented in chronological order. The presented data security frameworks only considered the security of mobile users' data created and manipulation on the mobile device in the *MCC* environment.

3.1.1. Energy efficient framework for integrity verification of storage services in MCC

Itani et al. [38] proposed an energy efficient framework for mobile devices to ensure the integrity of the mobile users' files/data stored on the cloud server using the concept of incremental cryptography and trusted computing [39,40]. The system design

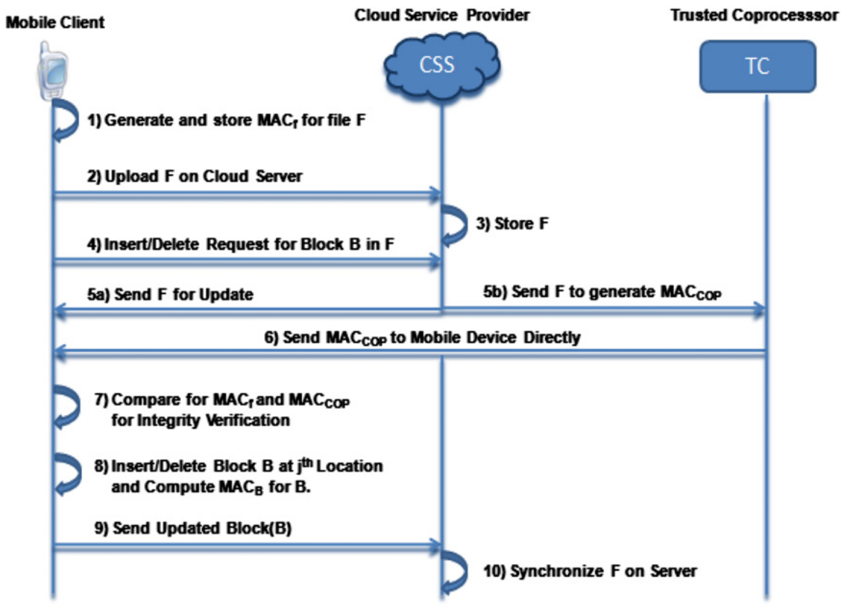
Mobile Client
Cloud Service Provider
CSS
Trusted Coprocessor
TC
1) Generate and store MAC$_f$ for file F
2) Upload F on Cloud Server
3) Store F
4) Insert/Delete Request for Block B in F
5a) Send F for Update
5b) Send F to generate MAC$_{COP}$
6) Send MAC$_{COP}$ to Mobile Device Directly
7) Compare for MAC$_f$ and MAC$_{COP}$ for Integrity Verification
8) Insert/Delete Block B at jth Location and Compute MAC$_B$ for B.
9) Send Updated Block(B)
10) Synchronize F on Server

Fig. 4. Interaction among the mobile client, cloud service provider and trusted coprocessor.

contains three main entities: (a) mobile client, (b) cloud service provider, and (c) trusted third party. The mobile client utilizes the storage services provided by the cloud service provider. The cloud service provider is responsible for managing, operating, and allocating the cloud resources efficiently. The trusted third party is responsible for configuring and installing the tamperproof coprocessors on the remote cloud. Each coprocessor is associated with multiple registered mobile clients. The coprocessor distributes Secret Key (SK) with associated mobile clients and generates a message authentication code on behalf of mobile clients. Interactions among the components are shown in Fig. 4.

The authors have discussed uploading, block insertion, block deletion, and integrity verification operations for files in the MCC environment. For uploading a file on cloud server, the mobile client generates an incremental Message Authentication Code (MAC_f) using SK shown in Eq. (1).

$$MAC_f = \sum_{i=1}^{k} HMAC\ (F_i, SK) \tag{1}$$

where k represents the total logical partition of file, F_i corresponds to the ith part of the file, and MAC_f represents the sum of increment message authentication codes. The mobile client stores MAC_f on local storage and uploads files on a cloud server.

At any time, the mobile client can perform insert, delete, and update operations on uploaded file(s). To insert or delete a block at the jth position of file, the mobile client requests a cloud server for the file. The cloud server transfers files to the mobile client as well as to the trusted coprocessor. The trusted coprocessor reconstructs and sends the MAC_{cop} to the mobile client. The mobile client receives the file and MAC_{cop} from the cloud server and trusted coprocessor respectively. The received MAC_{cop} is compared with the stored MAC_f for integrity verification. The same value of received and recalculated $MACs$ confirms the integrity of the file. After integrity verification, the mobile client inserts/deletes a block at the jth location of the file and reproduces MAC_f using inserted/deleted blocks, old MAC_f, and SK. To avoid the communication overhead, only the updated block with location information is uploaded on the cloud server for storage synchronization. The mobile client can verify the integrity of file(s) stored on the cloud server. The mobile client initiates the integrity verification process by sending a request to cloud server(s). On receiving the request, the cloud server(s) transfer files to the trusted coprocessor. The trusted coprocessor computes the incremental authentication code for each file and sends to the mobile client. The mobile client only needs to compare the stored MAC_f with received MAC_{cop} to verify the integrity of files.

Experimental results of the proposed framework show a 90% saving in processing and energy on the mobile device as compared to conventional techniques that generate MAC_f from scratch.

3.1.2. A framework for secure data service in MCC

Jia et al. [41] proposed a secure data service that outsources data and security management to cloud in trusted mode. The secure data service allows mobile users to move data and data sharing overhead to cloud without disclosing any information. There are three main entities involved in the proposed network model: (a) data sharer, (b) data owner, and (c) cloud service provider shown in Fig. 5. The data owner shares files and grants access privileges to the data sharer. The data owner and data sharer both utilize the cloud storage service to store and retrieve files.

The proxy re-encryption [42] and identity base encryption [43] schemes are used to achieve the secure data service. In the proxy re-encryption scheme, a semi trusted proxy transforms ciphertext encrypted with A's public key into another ciphertext encrypted with B's public key [44]. The identity based encryption scheme is based on bilinear mapping. A bilinear map can be defined as:

$$e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T. \tag{2}$$

Eq. (2) represents the efficient bilinear map having the properties of bilinearity, computability, and non-degeneracy [43]. The $\mathbb{G}_1$ and $\mathbb{G}_T$ are the multiplicative cyclic groups with prime order q and g is the generator of the $\mathbb{G}_1$. Two independent hash functions H_1 and H_2 are used in the proposed scheme shown in the following equation.

$$H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1, \qquad H_2 : \mathbb{G}_T \rightarrow \mathbb{G}_1. \tag{3}$$

The proposed secure data service works in six phases discussed in the following section:

(1) *Setup phase*: In this phase, system Master Key (MK) and system parameters ($\mathbb{G}_1, \mathbb{G}_T, g, g^s$) are generated, where $s \in \mathbb{Z}_q$ is randomly selected. Only the authority has information about the MK while system parameters are public and disseminated among mobile users.

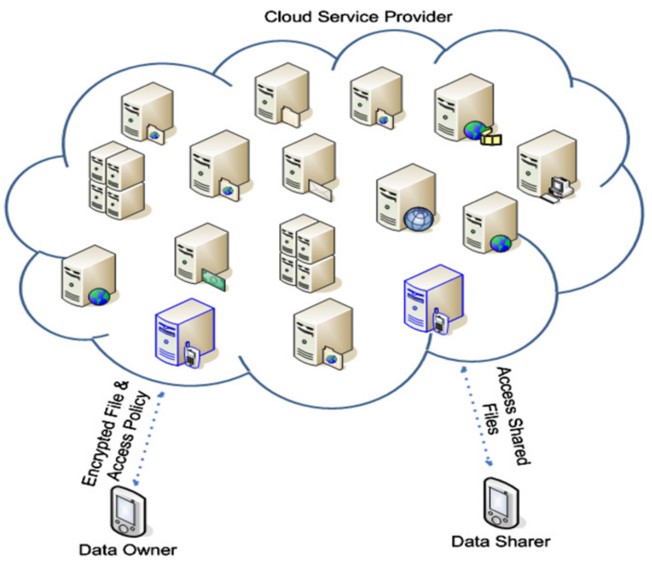

Fig. 5. Proposed framework's network model [41].

(2) *Key generation phase*: Mobile users have to register in the system to obtain a *SK* on the basis of mobile users' identity using *MK* and H_1 as illustrated in the following equation.

$$SK_{ID} = H_1(ID)^{MK}, \quad \text{where } ID \in \{0, 1\}^*. \tag{4}$$

(3) *Encryption phase*: Mobile user divides file into *n* chunks. Each chuck is represented with m_i and encrypted under owner identity (ID_{owner}) as a public key. The encryption under identity is referred as identity based encryption.

$$EF = \left(g^r, m_i \cdot e\left(g^s, H_1\left(ID_{owner}\right)^r\right)\right) \quad \text{where } 1 \le i \le n. \tag{5}$$

Here, $r \in \mathbb{Z}_q$ is selected randomly. The Encrypted File (*EF*) is uploaded on the cloud server.

(4) *Re-encryption key generation phase*: After uploading the *EF*, the mobile user generates the re-encryption keys for authorized users to access the file.

(5) *Re-encryption phase*: The re-encryption keys are sent to the cloud for encryption of the *EF* using the proxy re-encryption scheme. The proxy re-encryption scheme transforms ciphertext encrypted with owner's public key into a ciphertext encrypted with sharers' public keys.

(6) *Decryption phase*: The sharer requests the cloud server to obtain the re-encrypted file. The cloud verifies the validity by checking the re-encryption key for the sharer. If a key is found, the cloud server sends the corresponding re-encrypted file to the sharer. Due to the ciphertext transformation, the sharer can decrypt the file without the involvement of the data owner.

The proposed secure data service not only provides the data privacy but also fine-grained access control with the minimum cost of updating access policy and communication overheads.

3.1.3. A framework for secure storage services in MCC

Hsueh et al. [45] proposed a scheme for smart phones to ensure the security and integrity of mobile users' files stored on cloud server(s). The authors also introduced an authentication mechanism to authenticate the owner of the uploaded file on cloud. The proposed framework consists of four modules: (a) mobile device, (b) cloud service provider, (c) certification authority, and (d) telecommunication module shown in Fig. 6. Mobile device utilizes cloud services. The certification authority is responsible for authentication of mobile devices. The telecommunication module

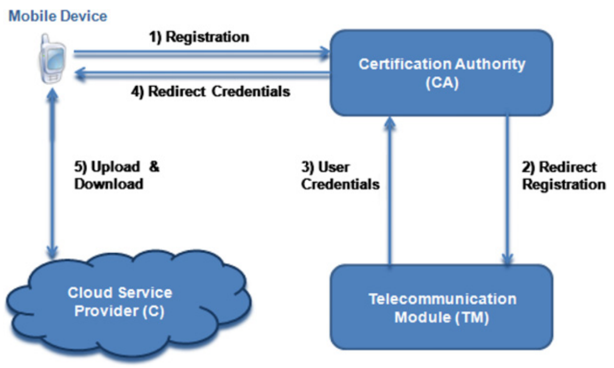

Fig. 6. Proposed framework [45].

generates and keeps track of mobile devices' passwords and related information to use cloud services. The authors suppose that *SK*, Public Key (*PK*), and Session Key (*SEK*) are securely distributed among mobile devices, the telecommunication module, and certification authority. To use the cloud services, the mobile user has to register with the telecommunication module through the certification authority. On successful registration, the telecommunication module issues a Password (*PWD*) for mobile device to use cloud resources. The registration request is represented as:

$$\begin{aligned} MD \to CA : E_{PK_T}E\,(MU, Num, TK)\,, \\ U_N, S_{SK_{MU}}\,(H\,(MU, Num))\,, H(MU, Num). \end{aligned} \tag{6}$$

The *MU* represents the mobile user's name, *Num* is the representation of the mobile user's number, *TK* is the combination of the *Num* and *PWD*, U_N depicts the randomly generated number for the proof of identity, *H* is a standard hash function, $E_{PK_{TM}}$ represents encryption with the *PK* of telecommunication module, and $S_{SK_{MU}}$ generates a signature for the mobile user using a cryptographic function on the passed value and *SK* of the mobile device. After receiving the message from the mobile device, the certification authority validates the authenticity of the message with the help of the received signature. If the message is from a valid user, the certification authority sends the following message to the telecommunication module.

$$\begin{aligned} CA \to TM : E_{PK_{TM}}\,(MU, Num, TK)\,, \\ U_N, S_{SK_{CA}}\,(H\,(MU, Num))\,. \end{aligned} \tag{7}$$

The telecommunication module authenticates the certification authority using received $S_{SK_{CA}}$. On successful authentication of the certification authority, the telecommunication module registers the mobile user by storing the mobile user's information in a local database. The stored information is used for verification of mobile users in the future. The telecommunication module is also responsible to generate *PWD* for the mobile user to access cloud resources. For secure delivery of *PWD* to a mobile device, the telecommunication module encrypts the mobile user's information using the mobile device's *PK*. The *PWD* is further encrypted with *TK* to ensure that only the authorized mobile user can decrypt the *PWD*. The telecommunication module forwards the encrypted information to the mobile device through the certification authority as shown below.

$$TM \to CA : E_{PK_{MU}}\,(MU, Num, U_N, E_{TK}\,(PWD)) \tag{8}$$

$$CA \to MD : E_{PK_{MU}}\,(MU, Num, U_N, E_{TK}\,(PWD))\,. \tag{9}$$

Mobile device encrypts file with *SEK* and uploads file along with *PWD*, *MU*, and $S_{SK_{MU}}$ on cloud.

$$\begin{aligned} MD \to C : PWD, MU, E_{SEK}\,(Data)\,, \\ S_{SK_{MU}}\,(H\,(MU \| SV \| E_{SEK}\,(Data))) \end{aligned} \tag{10}$$

where SV represents the secret value generated by the mobile device. The authors assume that SV is known to the mobile device, cloud, and telecommunication module. To download a file, the mobile device sends PWD, MU, and $H(MU \parallel SV)$ to cloud. Cloud regenerates the hash value using MU and SV. The newly computed hash value is compared with the received signature for authentication of the mobile device request. After successful authentication, cloud sends the encrypted file to the mobile device along with a signature.

$$C \rightarrow MD : E_{SEK}\,(Data)\,,\, H\,(E_{SEK}\,(Data) \parallel SV)\,. \tag{11}$$

The mobile device verifies the signature and decrypts the file using SEK. The authors also introduced a secure file sharing mechanism between two mobile users, say A and B. If user B wants to share a file with user A, user B has to inform user A about MU_B, Num_B, and PWD_B to access files on cloud. After downloading file, user A can authenticate the owner of the file on the basis of a signature stored with the encrypted file. The sharing mechanism reveals user B's secret information on A. User A can utilize the secret information to impersonate user B in the future.

3.1.4. A public provable data possession scheme for MCC

Yang et al. [46] extended the public provable data possession scheme proposed in [47] for a resource constrained mobile device that ensures the privacy, confidentiality, and integrity of mobile users' data stored on cloud. The system model consists of three main entities: (a) mobile end-user, (b) trusted third party, and (c) cloud storage service. The mobile end-user utilizes the cloud storage and may request data storage validation. Trusted Third Party (TTP) provides encryption, decryption, and authentication services for the mobile end-user to overcome the processing burden. The cloud Storage Service (CSS) contains an enormous amount of storage for mobile end-users. The CSS is also responsible for providing proof of data possession when requested by TTP or the mobile end-user. The proposed scheme uses Diffie–Hellman key exchange [48], bilinear mapping [49], and Merkle Hash Tree (MHT) [50]. Diffie–Hellman key exchange is used to securely distribute the symmetric key between two users without the involvement of a third party. A map $e\ :\ _1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ having properties of bilinearity, computability, and non-degeneracy is called a bilinear map. Where $\mathbb{G}_1$ is the gap Diffie–Hellman group, $\mathbb{G}_T$ is the cyclic multiplicative group of prime order p, and g is the generator of $\mathbb{G}_1$. MHT is constructed as a binary tree where leaf nodes contain the hash value of data blocks. The MHT is an efficient way to identify that the set of data blocks are unchanged or undamaged. During the verification, the verifier only needs to verify the root hash value of the tree. In the setup phase, TTP and mobile end-user share the symmetric key using a Diffie–Hellman key exchange process. The Diffie–Hellman key exchange process uses the private keys α and β of mobile-end user and TTP respectively. The key exchange process is shown in steps 1 and 2 of Fig. 7.

The mobile end-user encrypts the file with shared symmetric key ($g^{\alpha\beta}$). The encrypted file is delivered to the TTP. After receiving the file from the mobile end-user, TTP generates the pair of asymmetric keys (ek, dk) for encryption and decryption respectively. TTP encrypts the file using ek, divides the encrypted file into N small chunks, encodes each chunk with erasure codes, and constructs MHT. The leaf nodes of the MHT represent the hash values of the file's chunks. The TTP is also responsible for generating the hash of the root of MHT tree represented with $H(R)$, encrypting dk and $H(R)$ with the shared symmetric key, and delivering encrypted information to the mobile end-user. The mobile end-user signs $H(R)$ with private key α to generate a meta signature ($Sig_{SK}(H(R))$). The mobile end-user sends a meta signature back to TTP. The encrypted file is uploaded on a cloud

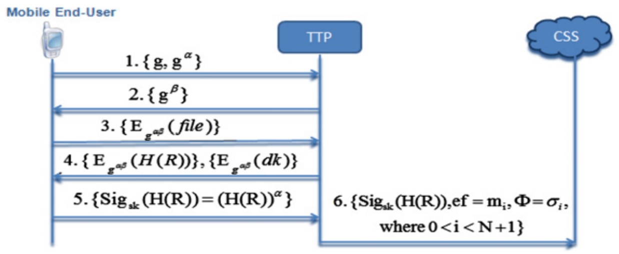
Mobile End-User
TTP
CSS
1. {g, g^α}
2. {g^β}
3. {E_{g^{αβ}}(file)}
4. {E_{g^{αβ}}(H(R))}, {E_{g^{αβ}}(dk)}
5. {Sig_sk(H(R)) = (H(R))^α}
6. {Sig_sk(H(R)), ef = m_i, Φ = σ_i, where 0 < i < N+1}

Fig. 7. Setup phase of proposed scheme.

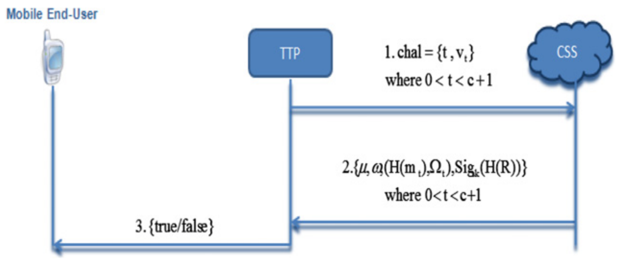
Mobile End-User
TTP
CSS
1. chal = {t, v_t} where 0 < t < c+1
2.{μ, ω(H(m_t), Ω_t), Sig_sk(H(R))} where 0<t<c+1
3. {true/false}

Fig. 8. Integrity verification phase of the proposed scheme.

server along with $Sig_{SK}(H(R))$ and a collection of file chunks' signatures (Φ). The Φ is evaluated using the following equation.

$$\Phi_i = [H(m_i)u^{m_i}]^{\beta}, \quad 1 \le i \le N \tag{12}$$

where u is randomly chosen from $\mathbb{G}_1$ and m_i is the ith chunk of the file.

In the integrity verification phase, the mobile end-user or TTP sends a challenge to CSS as shown in Fig. 8. The challenge is built by choosing c random variables in the range of 1 to N to construct a set T. For each $t \in T$, the corresponding $v_t \in \mathbb{Z}_p$ is randomly selected. The challenge (t, v_t) for each $t \in T$ is sent to the CSS. After receiving the challenge, CSS generates a proof of verification that includes $H(m_t)$ of the corresponding data block for every $t \in T$, μ, ω, and additional information (Ω_t) to rebuild the root $H(R)$. The μ and ω are calculated using following equations.

$$\mu = \sum_{t=1}^{c} v_t m_t \in \mathbb{Z}_p \tag{13}$$

$$\omega = \prod_{t=1}^{c} \Phi_t^{v_t} \in \mathbb{G}_1. \tag{14}$$

The generated proof is sent to TTP. The TTP reconstructs MHT on the basis of received $H(m_t)$ and Ω_t. The integrity verification is confirmed, if the bilinear mapping of the following equations satisfies the equality.

$$e\,(Sig_{sk}\,(H\,(R))\,,g) \overset{?}{=} e\;\,(H\,(R)\,,g^{\alpha}) \tag{15}$$

$$e\,(\omega, g^{\alpha}) \overset{?}{=} e\left(\prod_{t=1}^{c} (H\,(m_t)^{v_t}\,u^{\mu})\,, g^{\alpha\beta}\right). \tag{16}$$

In the secure file retrieval phase, the mobile end-user and TTP share SK through Diffie–Hellman key exchange. The mobile end-user encrypts dk with SK and sends to TTP. The TTP requests CSS for the encrypted file. The encrypted file is decoded and decrypted by TTP to get the original file. Finally, TTP sends the decrypted file to the mobile-end user through the secure channel.

In the public provable data possession scheme, TTP sends an encrypted and encoded file to CSS that ensures the privacy and confidentiality of the file. The signature structure of MHT helps to ensure the integrity of file with minimum storage and communication overhead.

3.1.5. Lightweight and compromise resilient storage outsourcing in MCC

Ren et al. [29] proposed three schemes to ensure the confidentiality and integrity of the users' files stored on cloud. The files are assumed to be created and operated only on a mobile device. The files may be stored on single or multiple cloud servers. The authors assume the cloud servers as distrusted nodes, the mobile device as semi-trusted in case of storage, and trusted in case of computation. The mobile device is considered semi-trusted in terms of storage due to the fact that the loss of the device exposes all the storage information. The mobile device is considered trusted in case of computation for the reason that a mobile user may take precautionary measures to hide the execution environment of mobile device from malicious activities. In each scheme, the mobile device is responsible for encryption, decryption, and integrity verification. The authors discuss secure uploading and downloading of a file on cloud server(s) in each scheme.

3.1.5.1. Encryption based scheme (*EnS*). When a user wants to upload a file on a cloud server through a mobile device, the user has to provide a *PWD*. Mobile device generates Encryption Key (*EK*) and Integrity Key (*IK*) by applying standard Hash function (*H*) on concatenation of File Name (*FN*), File Size (*FS*), and *PWD* depicted in the following equations.

$$EK = H(PWD \| FN \| FS) \tag{17}$$

$$IK = H(FN \| PWD \| FS). \tag{18}$$

Mobile device encrypts file with *EK* for confidentiality and generates *MAC* using file and *IK* for authentication.

$$EF = E_{EK}(file) \tag{19}$$

$$MAC = HMAC(file, IK). \tag{20}$$

Mobile device uploads *EF*, $H(FN)$, and *MAC* on the cloud server, deletes *IK* and *EK*, and stores *FN* in the local file table. While downloading a file, mobile device computes $H(FN)$ and transfers the hash value to the cloud server. The cloud server looks for corresponding *EF* and *MAC* with the help of received $H(FN)$. If a file is found, the cloud server sends *EF* and *MAC* to the mobile device. The mobile device prompts the user to enter a *PWD* for the file to regenerate *IK* and *EK* for integrity verification.

$$EK = H(PWD \| FN \| FS) \tag{21}$$

$$IK = H(FN \| PWD \| FS). \tag{22}$$

Afterwards, the mobile device decrypts the *EF* using regenerated *EK* to get the original file. The mobile device also regenerates *MAC* using *FN* and regenerated *IK*.

$$Decrypted\ File\ (file) = D_{EK}(EF) \tag{23}$$

$$MAC = HMAC(file, IK). \tag{24}$$

The same value of received and recalculated *MACs* confirms the integrity of the file.

3.1.5.2. Coding based scheme (*CoS*). *CoS* reduces the computation overhead of encryption imposed by *EnS* using a lightweight computation operation to preserve the privacy of the users' files. When a mobile user wants to upload a file on the cloud server through a mobile device, the user has to provide a *PWD*. The mobile device divides the file into *d* parts such that each part has *t* chunks and each chunk has *n* bits. Suppose the mobile user wants to store a file on *d* cloud server(s). The mobile device generates coding vector (α) by applying the recursive hash function on a concatenation of *PWD*, *FN*, and *FS* for secrecy as stated in the equations below.

$$\alpha_i = H^i(PWD \| FN \| FS), \quad \text{where } 1 \le i \le t. \tag{25}$$

H^i is evaluated as:

$$H^1(x) = H(x), \qquad H^i(x) = H\left(H^{i-1}(x)\right) \quad \text{where } 2 \le i \le t. \tag{26}$$

The mobile device generates *IK* by applying a hash function on the concatenation of α.

$$IK = H(\alpha_1 \| \alpha_2 \| \alpha_3 \cdots \| \alpha_t). \tag{27}$$

The mobile device uses α to produce Secrecy Code (*SC*) for each part of the file to achieve confidentiality.

$$SC[j] = \left(\sum_{i=1}^{t} \alpha_i * F[i][j]\right) \quad \text{where } 1 \le j \le d \tag{28}$$

where $F[i][j]$ represents the *j*th part of the file. The mobile device constructs the *MAC* using the file and *IK*.

$$MAC = HMAC(file, IK). \tag{29}$$

Afterwards, the mobile device uploads $SC[j]$, $H(FN + j)$, and *MAC* on cloud server (CS_j), where *j* is randomly selected in the range of 1 to *d*. The mobile device saves *FN* on local table and deletes α and *IK*.

While downloading the file from cloud server(s), the mobile device regenerates $H(FN + j)$ and sends it to the corresponding CS_j. The CS_j looks for *SC* and *MAC* in local storage using the received $H(FN + j)$. If a record is found, the cloud server sends the corresponding *SC* and *MAC* to the mobile device. The mobile device prompts the user to enter a *PWD* for the file to reproduce α and *IK*.

$$\alpha_i = H^i(PWD \| FN \| FS) \quad \text{where } 1 \le i \le t \tag{30}$$

$$IK = H(\alpha_1 \| \alpha_2 \| \alpha_3 \cdots \| \alpha_t). \tag{31}$$

The mobile device decodes the file by multiplying *SC* with the inverse of α. The multiplication of *SC* with the inverse of α produces the original file.

$$file = \left(\alpha_i^{-1} * SC[j]\right) \quad \text{where } 1 \le i \le t,\ 1 \le j \le d. \tag{32}$$

After getting the original file, the mobile device regenerates *MAC* using the decrypted file and regenerated *IK*.

$$MAC = HMAC(file, IK). \tag{33}$$

The same value of received and recalculated *MACs* confirms the integrity of the file.

3.1.5.3. Sharing based scheme (*ShS*). *ShS* is more energy efficient as compared to *EnS* and *CoS*. The *ShS* introduces a simple exclusive-OR (*xor*) based secret sharing mechanism that requires less computation power on the device side. When the user wants to upload a file on the cloud server through a mobile device, the user has to provide a *PWD*. The mobile device generates *IK* using a hash function on *FN*, *FS*, and *PWD*.

$$IK = H(FN \| PWD \| FS). \tag{34}$$

The mobile device generates $(d - 1)$ random files/shares donated by $RS[j]$, where *j* is in the range of 1 to $(d - 1)$. The *d*th share is produced with the help of two *xor* operations. The first *xor* operation is performed on $(d - 1)$ randomly generated files/shares to produce Accumulative Results (*AR*). The second *xor* operation is performed on the original file and *AR* as shown below.

$$AR = \left(\bigoplus_{i=1}^{d-1} RS[i]\right) \tag{35}$$

$$RS[d] = AR \bigoplus file. \tag{36}$$

The mobile device further generates MAC using the file and IK to achieve integrity.

$$MAC = HMAC(file, IK). \tag{37}$$

The mobile device uploads $RS[i]$, $H(FN+i)$, and MAC on CS_i, where i is randomly selected in the range of 1 to d. The mobile device saves FN and deletes IK. When the user wants to download a file from CS through the mobile device, the mobile device computes $H(FN+i)$ and sends to the corresponding CS_i. The corresponding CS_i looks for $RS[i]$ and MAC with the help of the received $H(FN+i)$. If a record is found, CS_i sends the corresponding $RS[i]$ and MAC to the mobile device. The mobile device prompts the user to enter a PWD for the file to reproduce IK.

$$IK = H(FN \| PWD \| FS). \tag{38}$$

Now the mobile device decodes the secrecy code by simply applying xor on all SC.

$$file = \left(\bigoplus_{i=1}^{d} RS[i] \right). \tag{39}$$

After getting the original file, the mobile device computes MAC for integrity checking using the original file and IK.

$$MAC = HMAC(file, IK). \tag{40}$$

The same value of received and recalculated $MACs$ confirms the integrity of the file.

The authors further analyzed each scheme for confidentially and integrity in an environment where cloud servers were fully distrusted and encrypted files were stored on single or multiple cloud server(s). The authors came to conclusion that the probability of guessing the EK and IK is negligible. Other malicious activities to obtain EK and IK either generate a large amount of traffic or require a lot of processing. Both activities can be identified by an intrusion detection system installed on a mobile device.

3.1.6. *A security framework for efficient and secure data storage services in MCC*

Zhou and Huang [51] proposed a privacy preserving framework called Privacy Preserving Cipher Policy Attribute-Based Encryption (PP-CP-ABE) for lightweight mobile devices. The proposed scheme offloads the processing and storage intensive encryption and decryption operations on cloud without revealing any information about data contents and security key. The authors also proposed an attribute based data storage system that provides cryptographic access control to overcome the communication and storage overhead for data management on mobile devices as well as on cloud. Here, we limit our discussion to PP-CP-ABE. The architecture of the proposed scheme contains four components: (a) data owner, (b) encryption service provider, (c) decryption service provider, and (d) storage service provider shown in Fig. 9.

The Data Owner (DO) can be a mobile device or a sensor that may request to store and retrieve data from cloud. To increase the processing capability of DO, a major portion of encryption and decryption operations are outsourced on cloud. Encryption Service Provider (ESP) encrypts the file for DO without having knowledge about the security keys. Decryption Service Provider (DSP) decrypts the file for DO without getting any information about data contents. The encrypted data is stored on a storage service provider. The authors assume that Trusted Authority (TA) is responsible for generating and distributing keys among DOs.

The PP-CP-ABE is based on bilinear mapping (discussed in Section 3.1.2) [52], access policy tree, and secret sharing scheme [53]. Access policy tree is constructed with the help of internal and leaf nodes. The leaf nodes represent the attributes associated with DO while internal node represents the logic gates (e.g. AND or OR). The

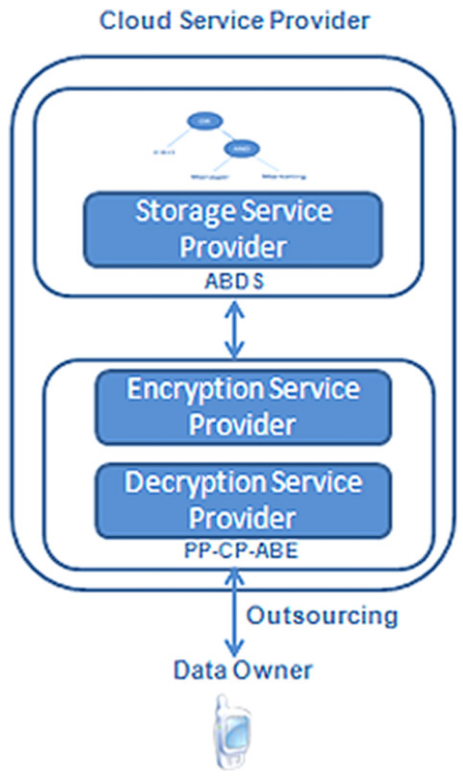

Fig. 9. Architecture of proposed framework [51].

secret sharing scheme divides the secret into n shares where any t shares can reconstruct the secret.

In the setup phase, TA randomly selects $\alpha, \beta \in \mathbb{Z}_p$ and generates a bilinear map on $\mathbb{G}_1$. The TA constructs the PK known to everyone and MK only known to TA.

$$PK = (\mathbb{G}_1, g, h = g^{\beta}, f = g^{1/\beta}, e(g, g)^{\alpha}) \tag{41}$$

$$MK = (\beta, g^{\alpha}). \tag{42}$$

Users need to get registered with TA to get private keys. The private key is generated on the basis of user's attributes.

$$SK = \left\{ D = g^{\frac{(\alpha+r)}{\beta}}, \forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j} \right\} \tag{43}$$

where $r \in \mathbb{Z}_p$, S represents user attributes, $r_j \in \mathbb{Z}_p$, and $j \in S$.

DO needs to specify the Data Access Tree (DAT) to outsource encryption operations. The DAT is divided into two sub trees DAT_{DO} and DAT_{ESP} to preserve data security while offloading the computation. DAT_{ESP} is cloud controlled data access policy and DAT_{DO} is DO controlled data access policy. The original DAT can be reconstructed as follows:

$$DAT = DAT_{DO} \wedge DAT_{ESP} \tag{44}$$

where $\wedge$ represents a logical AND gate and depends on the root node of DAT. The DAT_{DO} is dealing with a small number of attributes to reduce computational overhead on the device. Normally, DAT_{DO} contains one attribute. The DO randomly creates a one degree polynomial ($q_r(x)$) and generates secrets $s = q_r(0)$, $s1 = q_r(1)$, and $s2 = q_r(2)$. The DO sends DAT_{ESP} and $s1$ to ESP. On successful reception of DAT_{ESP}, ESP generates Temporal Cipher (TC) on the basis of received information as depicted in the following equations.

$$TC_{ESP} = \left\{ \forall y \in Y_{ESP} : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)} \right\} \tag{45}$$

$$q_y(0) = q_{parent(y)} \times (index(y)),$$
$$\text{where } q_{root}(0) = s1 \text{ in case of } DAT_{ESP} \tag{46}$$

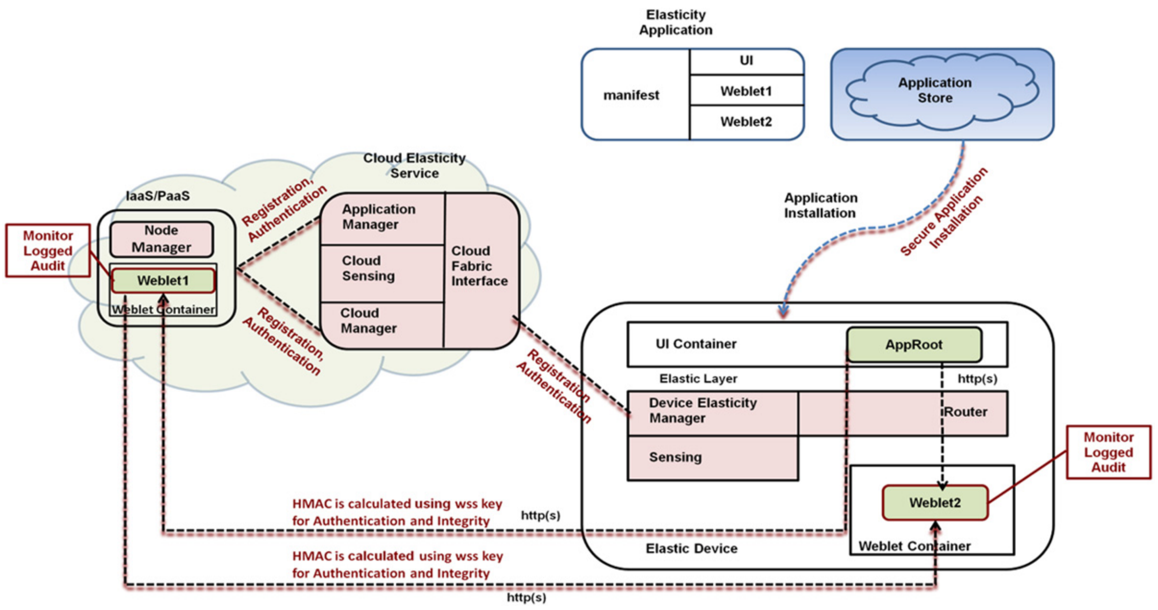

Fig. 10. Reference architecture of elastic application [34].

where Y_{ESP} represents the set of leaf nodes in DAT_{ESP}, $att(y)$ returns the attributes associated with the leaf node y, and $index(y)$ returns the unique index associated with each node. Meanwhile, DO completes the encryption process using s and $s2$.

$$TC_{DO} = \left\{\forall y \in Y_{DO} : C_y = g^{q_y(0)}, C'_y = H\,(att\,(y))^{q_y(0)}\right\} \tag{47}$$

$$C = Me(g, g)^{\alpha s}, \qquad C = h^s \tag{48}$$

where M is a message and e represents the bilinear mapping. DO sends TC_{DO}, $\hat{C}$, and C to ESP. The ESP generates the ciphertext on the basis of received information described in Eq. (49).

$$CT = \Big\{DAT = DAT_{ESP} \wedge DAT_{DO};\ \hat{C} = Me(g, g)^{\alpha s};$$
$$C = h^s;\ \forall y \in Y_{DO} \bigcup DAT_{ESP}\ C_y = g^{q_y(0)},$$
$$C'_y = H\,(att\,(y))^{q_y(0)}\Big\}. \tag{49}$$

To offload computation for decryption without revealing private key information, DO blinds the private key with the help of random number $t \in \mathbb{Z}_p$ as shown in Eqs. (50) and (51).

$$D^t = g^{\frac{t(\alpha+r)}{\beta}} \tag{50}$$

$$SK_B = \left\{D_B = g^{\frac{t(\alpha+r)}{\beta}}, \forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j}\right\}. \tag{51}$$

DSP uses the blinded key on encrypted file to generate a raw file. DO converts the raw file into the original file with acceptable processing and storage overhead. Interested readers may refer to [51] to see the detail of decryption process.

ESP works on DAT_{ESP} and $s1$ during encryption. For a given one degree polynomial $q(x)$, secrets s and $s2$ are information-theoretically secure, if $s1$ is known [54]. DSP works on a blinded key for decryption to produces a raw file. DO converts the raw file into the original file with minimum effort. Therefore, processing intensive encryption and decryption operations are outsourced on cloud without revealing any data contents during encryption and security keys during decryption.

All the frameworks discussed under the category of data security frameworks have been presented in Table 1 chronologically. Each security framework has been evaluated with reference to evaluation criteria discussed in Section 2.1.

3.2. Application security frameworks

The countermeasures for application security frameworks are presented in chronological order. The presented application security frameworks considered the security of the mobile application or mobile application model in the *MCC* environment.

3.2.1. Securing elastic application on mobile device for cloud computing

Zhang et al. [34] proposed a model for securing an elastic mobile application in a cloud environment. An elastic application [55] consists of: (a) one or more weblet(s), (b) user interface, and (c) manifest. Weblets are independent software modules that can run on a device or cloud and expose an interface for communication with other weblets or user interfaces. The manifest is a static *XML* file that contains metadata about an application e.g. digital signature, maximum weblet instances launched on mobile and cloud, and processing power requirements. The detail of the model is shown in Fig. 10.

The main component on the device side is the Device Elasticity Manager (*DEM*) that configures the application at launch time and makes configuration adjustments at runtime. The router module provides an interface between weblet and user to achieve the transparency between user interface and weblet location. The router module has complete information about each weblet execution location. When the weblet is migrated, the router module is informed about the update in location. The sensing module keeps track of device utilization and shares the information with *DEM*. The *DEM* uses device utilization information to take a decision of whether weblet(s) should be launched on a device or moved to cloud. On the cloud side, the main component is the Cloud Elasticity Service (*CES*) that consists of: (a) cloud manager, (b) application manager, and (c) sensing information collection. The responsibility of the cloud manager is to allocate cloud resources for the execution of weblet(s) and release them after the successful execution of weblet(s). The cloud manager maintains the resource's usage information like memory used, bandwidth consumed, and computation conducted by each weblet. The application manager installs and maintains the elastic application on behalf of the elastic device. The sensing module collects operational data on cloud. The cloud manager utilizes operational data for tracking usage. The sensing

Table 1
Comparison of evaluated data security frameworks.

	Basic theory	Data protection		Data integrity	Authentication	Scalability	Assumption (AP)		Data access
		ProDCMD	ProDCMC				AP-1	AP-2	
Itani et al. [38]	Incremental message authentication code	No	No	Yes	No	Moderate	Coprocessor is trusted	Coprocessor is tamperproof	–
Jia et al. [41]	Proxy re-encryption (PRE) scheme and identity base encryption (IDE) scheme	Yes	No	No	No	Highly scalable	Cloud server is semi trusted and assume to be always online	Adversary cannot hold re-encryption key	Automated
Hsueh et al. [45]	Standard cryptography functions	Yes	No	Yes	Yes	Moderate	Keys are securely distributed	Secret values are securely distributed.	Automated
Yang et al. [46]	Diffie–Hellman key exchange, bilinear mapping, and merkle hash tree	Yes	No	Yes	Yes	Moderate	Communication channel is secured	*TTP* is trust	–
Ren et al. [29] (EnS)	Standard cryptography functions	Yes	No	Yes	Yes	Highly scalable	Mobile device is semi trusted, cloud servers are distrusted	Communication channel is secured	Semi automated
Ren et al. [29] (CoS)	Coding vector	Yes	No	Yes	Yes	Highly scalable	Mobile device is semi trusted, cloud servers are distrusted	Communication channel is secured	Semi automated
Ren et al. [29] (ShS)	Exclusive-OR	Yes	No	Yes	Yes	Highly scalable	Mobile device is semi trusted, cloud servers are distrusted	Communication channel is secured	Semi automated
Zhou and Huang [51]	Bilinear pairing, access policy tree, and secret sharing scheme	Yes	No	No	No	Highly scalable	Cloud service provider is semi trusted	Security algorithms are secured	Automated

module is also accountable for monitoring the cloud resources for failure detection and resource availability. Cloud fabric interface is a web service provided by *CES* for mobile applications. Weblet runtime environment is hosted on each cloud node (referred as weblet container) that is responsible for the execution of weblet. The node manager is deployed on each cloud node to look after resource usage (e.g. memory, *CPU*) of a particular node (server) within cloud. The collected resource's utilization information is shared with application manager and cloud manager for further actions.

The authors have also addressed various security aspects of the elastic mobile application model. For secure installation of an application, the developer signed a *SHA*1 hash value for each weblet and stored it in a java like application package. When a user installs the application, the installer re-computes and compares the hash value for integrity verification. After successful verification, the installer registers the application with *DEM*. The *DEM* keeps track of all the applications that require elasticity manager support. As an installation option, part of the application can be installed on *CES* with the help of the application manager. Only registered and authenticated users can install the application on *CES*. The authors also introduced a mechanism that enables the weblet to authenticate another weblet of the same application. The *DEM* generates a weblet session key and weblet session secret. The keys are shared among all the weblets that belong to same application at launch time. During the authentication process, the weblet generates a Hash-based Message Authentication Code (*HMAC*) on the basis of nonce, source weblet *ID*, destination weblet *ID*, and weblet session secret. The original message and *HMAC* value are sent to the destination weblet. The destination weblet re-computes the *HMAC* using received message and own session secret key. The same value of received *HMAC* and re-computed *HMAC* confirms the authenticity of the weblet. Weblets can migrate between cloud and device. Usually *DEM* starts the migration process due to (a) increase in battery life of the device or (b) removal of an intensive execution burden from the device. For secure migration, *DEM* sends a request to the weblet to stop further processing. The weblet saves running state including session secret and returns to *DEM*. The *DEM* sends the request to the cloud manager through the cloud fabric interface for migration. The cloud manager allocates the resources to the migrated weblet and starts the execution from the last saved state. The weblet's new location information is propagated among all weblets of the application through *DEM*. The weblets running on cloud need authorization to access external web services before downloading user data. The authorization is only possible if the weblet has the capability to authenticate the user. Authentication can be achieved using different methods. In the shared user credential approach, the weblet may request the user obtains credentials (user name, password, or digital signature) for authorization. The shared user credential is a simple approach but involves risk. In the shared session information approach, weblet(s) running on a device authenticate from the web server and share the received application session key and application session secret among weblets for further communication with external web services. The shared session information approach shares only the session key that is valid for short time period. The third approach says that the weblet requiring external web service access should be executed only on the device. The proposed security framework covers secure installation of elastic application, authentication, secure migration, and authorization of weblets.

3.2.2. *A lightweight dynamic credential generation mechanism for user identity protection in MCC*

In a cloud computing environment, users' identity is mainly verified using digital credential e.g. a password or digital certificate. An adversary may hack the credential's information to impersonate the legal users in a cloud environment. The problem becomes more sophisticated in a mobile cloud environment due to the computational limitation of mobile device of running sophisticated security software. Xiao and Gong [56] proposed a lightweight algorithm for an *MCC* environment to generate the automatic dynamic credentials with the mutual coordination of mobile devices and cloud. The automatic generated credentials protect mobile users from an adversary. The proposed scheme changes the dynamic credential constantly on the basis of user and cloud communication. The exchanged data between cloud and user are transformed into dynamic secrets. If a message M_i is sent by the mobile user, an update in the user's dynamic secret (S_u) is done by applying an *xor* operation on the sent message shown below.

$$S_U = S_U \oplus M_i. \tag{52}$$

If a message M_i is sent by cloud, the update in cloud's dynamic secret (S_u) is done by applying a *xor* operation on a sent message as depicted in the following equation.

$$S_C = S_C \oplus M_i. \tag{53}$$

On each dynamic secrets update, there is an increase in packet counter ($N = N + 1$) as well. Dynamic credentials (S) are updated, when the mobile user requests for new data channel from the base station or user-cloud packets exchanged have been increased to a threshold ($N_{threshold}$) value. The new data channel is required, when mobile users switch between base stations or start new communication. The $N_{threshold}$ depends on how frequently the user wants to change the credential. The S is updated on the basis of S, S_u, and S_c illustrated in Eq. (54).

$$S = S \oplus H\,(S_U \parallel S_C) \tag{54}$$

where $\parallel$ represents a concatenation operation and H represents a standard hash functions, e.g. *SHA* or *MD*5. After updating the dynamic credential the values of S_u, S_c, and N are set to zero.

The frequent update in dynamic credential makes the recovery of S difficult for the adversary. Suppose an adversary keeps the information of S at time t_0 and plans for an attack at time t_1, where $t_1 > t_0$. The adversary must track all the communication of the user-cloud between t_0 and t_1 for accurate calculation of S at t_1. The possibility of attack seems to be impractical due to user mobility, unreliability of wireless communication, and resource limitation. Even the loss of a single packet between t_0 and t_1 makes the recovery of S difficult for an adversary.

3.2.3. *In-device spatial cloaking mechanism for privacy protection in MCC*

Spatial cloaking is used to protect the privacy of a mobile user while using the location-based services. Wang and Wang [57] proposed a top down spatial cloaking mechanism with and without optimization that utilizes cloud resources to provide a scalable, efficient, and improved privacy preserving framework for location-based services. The architecture of the proposed scheme is shown in Fig. 11.

The mobile client is responsible for generating the clocked region with the help of hierarchical decomposing of spatial space into h levels, where each level has 4^h grid cells [58]. The root at level zero covers the entire system area. In level one, the root is divided into four child grid cells. Level two is constructed by subdividing the each child grid cell of level one into four sub-child grid cells. The processing of subdivision is dependent on h. Two cells are considered to be the horizontal neighbors (C_H) of each other, if both cells have same parent and reside in the same row. Two cells are considered to be the vertical neighbors (C_V) of each other, if both cells have same parent and reside in the same column.

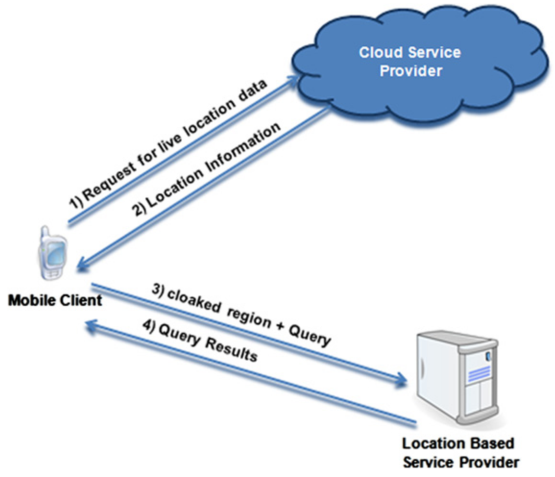
Cloud Service Provider
1) Request for live location data
2) Location Information
Mobile Client
3) cloaked region + Query
4) Query Results
Location Based Service Provider

Fig. 11. Architecture of in-device spatial cloaking in a cloud environment [57].

The cloud service provider maintains the updated location information of mobile clients in each grid cell. The cloud service provider is also responsible for maintaining the historical data about each grid cell. The location based service provider provides location-based services to the mobile client.

To provide location privacy while using the location-based services, the mobile client requests the cloud service provider for the live number of users in four child grid cells according to the mobile user's current grid cell position (C). If the live number of users in the child grid cell containing the request issuer (C_C) is less than threshold (k), the total number of live users is evaluated using the following equation.

$$\begin{aligned} Sum\ (S) &= Live\ No\ of\ Users\ in\ C_C \\ &\quad + Live\ No\ of\ Users\ (C_H \parallel C_V). \end{aligned} \tag{55}$$

If the S is greater than k, the generalized spatial region is generated on the basis of C_C and (C_H or C_V) any one having a live number of users less than k. The condition imposed on C_H or C_V is to anonymize the cell where the request issuer is located. If S is less than k, the current grid cell is considered as a generalized spatial region. If the C_C containing the request issuer has more than k live users, C_C becomes the new current cell. The same process is repeated on the new current cell until the bottom level of grid is reached or the child grid cell is found having a live number of users less than k.

The mobile client requests the cloud service provider to get a live number of users in a particular grid cell. The process of getting the live number of users induces a delay and communication overhead. The authors introduce a cloaked algorithm with optimization that overcomes the delay and communication overhead. The algorithm works on historical data instead of communicating with cloud to get the live number of users' information in any region. The historical data is maintained by cloud. The historical data contains the historical information about the live number of users in each grid cell. The correctness of the algorithm is dependent on the historical lower bond of the live number of mobile users in each grid cell. The lower bound collected information is shared with the mobile users at startup time with some initial communication cost.

The authors have evaluated the performance of the proposed cloaking technique with a traditional cloaking mechanism called Casper [59] on the basis of cloaking area, cloaking time, and communication cost. The experimental results depict that the proposed algorithms achieved a significant improvement as compared to the traditional cloaking algorithm.

3.2.4. MobiCloud: a secure cloud framework for mobile computing and communication

Huan et al. [60] proposed a new *MCC* framework that not only provides conventional computation services but also improves the functionality of the Mobile Adhoc Network (*MANET*) in terms of risk management, trust management, and secure routing. The proposed framework alters traditional *MANET* into a new service oriented model. The new service oriented model considers each mobile node as a service node. The service node can provide or consume services depending on the capability of the node. The services may be sensing services, storage services, or computation services. To overcome the uncertainty caused by the mobility, one or more Extended Semi Shadow Images (*ESSI*) are mirrored on cloud. The *ESSI* can be a partial clone, an exact clone, or an image of device having extended functionality. The *ESSI* and mobile node communicate with each other through secure connections like *SSL*, *IPSec*, etc. The architecture of MobiCloud is shown in Fig. 12.

Software agent allows mobile users to link the mobile device and cloud services. The same software agent can run on a mobile device as well as on a cloud platform correspondingly. The mobile device can run multiple software agents under the supervision of an application manager to access multiple cloud services or *MANETs*. The sensor manager on the device side utilizes sensing capabilities of the mobile device to get information about the device status (battery state, location information, and processor utilization) and as well as neighboring mobile node information (fellow identity, link strength, and remaining energy). On the server side, the MobiCloud application interface module provides services that are being utilized by mobile devices. The MobiCloud application interface module is also responsible to provide interfaces for: (a) MobiCloud Virtual Trust and Provisioning Domain manager (*VTaPD*) module and (b) resource and application manager module. Programmable routers [61] are used to provide a network virtualization service called *VTaPD*. The *VTaPD* are used to create multiple virtual domains for isolation of information flow and data access control. Each *VTaPD* is associated with multiple software agents that may belong to different *ESSI*. A node belonging to a particular *VTaPD* has the complete routing information of the residing *VTaPD*. Data access isolation is achieved with the help of implicit filtering and explicit permission [44]. The node manager is accountable for loading and unloading the software agents on *ESSI*. The *VTaPD* manager module and trusted management server module instruct the MobiCloud resources and application manager module for the creation of new *VTaPD* to provide security as a service. The *VTaPD* manager collects the sensing information from the device to make a decision about intrusion detection and risk management. The trust management server module is also responsible to handle the key management, data access management, and user-centric identity management. The authors introduce a novel approach called attribute based key management that is an extension of identity based cryptography [44]. The attribute based key management uses multiple attributes to identify an entity. The attributes are considered as public keys and corresponding paired private keys are generated by the trusted authority. The trust management server module distributes private keys among mobile users securely. The private keys are generated from descriptive terms instead of large prime numbers used in traditional key generation algorithms. The new private key generation mechanism requires less processing time to manage the security policies. The trust management server module is responsible for user-centric identity management through the authors' proposed attribute based identity management scheme. The attribute based identity management scheme defines Point of Network Presence (*PoNP*). The line radiating from the *PoNP* shows the relationship of mobile user's to various counterparties. Each *PoNP* is a combination of

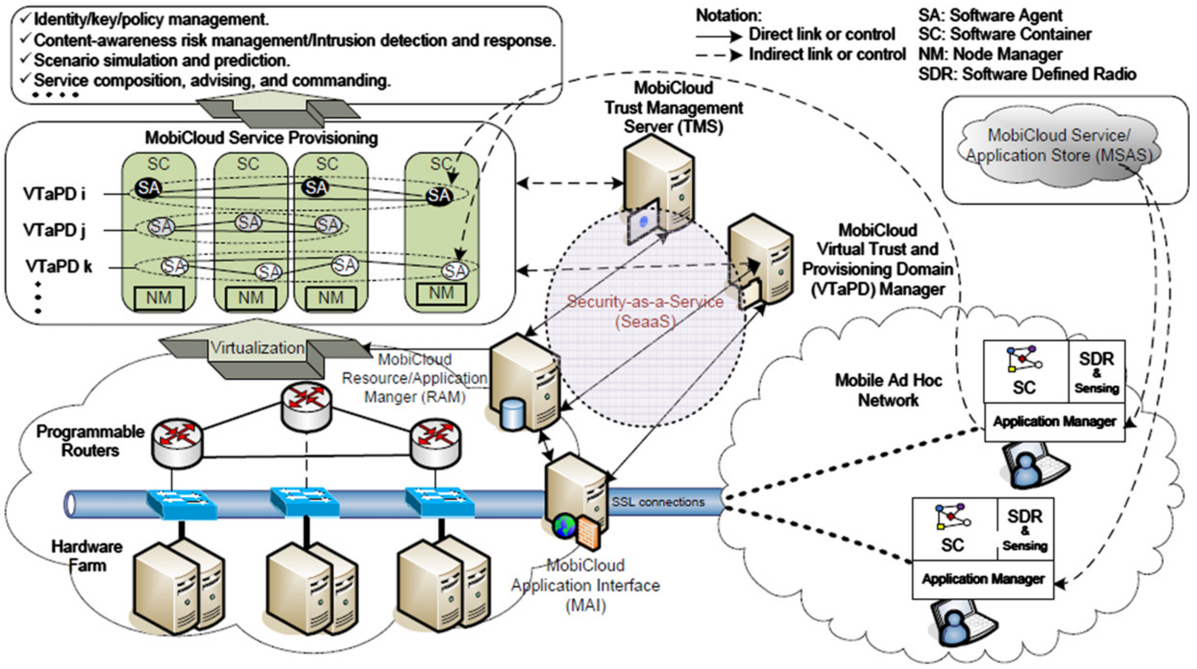
✓Identity/key/policy management.
✓Content-awareness risk management/Intrusion detection and response.
✓Scenario simulation and prediction.
✓Service composition, advising, and commanding.
Notation:
Direct link or control
Indirect link or control
SA: Software Agent
SC: Software Container
NM: Node Manager
SDR: Software Defined Radio
MobiCloud Trust Management Server (TMS)
MobiCloud Service/ Application Store (MSAS)
MobiCloud Service Provisioning
SC
SA
VTaPD i
VTaPD j
VTaPD k
NM
MobiCloud Virtual Trust and Provisioning Domain (VTaPD) Manager
Security-as-a-Service (SeaaS)
Virtualization
MobiCloud Resource/Application Manger (RAM)
Programmable Routers
Mobile Ad Hoc Network
SDR & Sensing
Application Manager
SSL connections
Hardware Farm
MobiCloud Application Interface (MAI)

Fig. 12. Reference model of MobiCloud [60].

type, value, and attributes. The type consists of: (a) identity issuer, (b) private key issuer, and (c) validation period. The *PoNP* may have multiple attributes. Each attribute is the combination of type and value. The default *PoNP* is associated with each individual having a unique value. The uniqueness can be achieved by applying a publicly known hash function on some uniquely identifiable attribute of the mobile user (e.g. passport number, email address or driver license identity). The trust management server module may use multiple *PoNPs* or attributes to define the publicly known native identities for the device. The generated identities are used for authentication, authorization, and access control. The MobiCloud service and application store module contains a repository of software agents and applications. The MobiCloud service and application store module installs the required software agents or applications through the MobiCloud application interface module when service composition is needed.

MobiCloud architecture is beneficial for *MANET* in term of security, risk assessment, location-based services, network status monitoring, and context aware routing. The architecture not only extends and augments the functionality of *MANETs* but is also helpful in predicting the future *MANET* situations for decision making by simulating different scenarios on cloud using historical sensed data.

3.2.5. Authentication framework for MCC

Chow et al. [62] proposed a policy based cloud authentication platform that addresses the client device authentication issue in a simple and flexible manner. The proposed system utilizes the TrustedCube to manage the authentication infrastructure and implicit authentication to translate user behaviors into score. Implicit authentication is also referred as behavioral authentication. The behavioral authentication uses habits instead of belonging (e.g. memorize data and biometric) to authenticate users. With the help of a statistical model, probabilistic authentication scores are assigned to client devices on the basis of observed behaviors. The proposed authentication framework compares threshold values with a user authentication score to identify whether the device is in the hands of a legitimate user or not. The threshold value is dependent on the type of application. There are four main components of the system model shown Fig. 13.

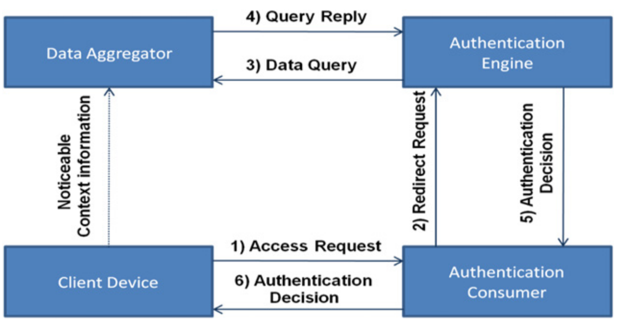

Fig. 13. System model diagram [62].

The client device generates the noticeable context and action like *SMS*, *MMS*, phone calls history, Internet browsing, network information, location information, and phone information. The client device keeps the generated data in a local cache until the data is collected by data aggregator. The authentication engine extracts data from the data aggregator and corresponding authentication policies from authentication consumer to authenticate client device. The authentication consumer enforces authentication policies on the basis of the client device request. Finally, the authentication consumer responds to the client's request on the basis of results received from the authentication engine.

The authentication process starts when the authentication consumer receives the access request generated by the client device. Before forwarding the request to the authentication engine, the authentication consumer registers a policy with the authentication engine. The policy consists of three parts: (a) access request (e.g. money transaction, webpage access request), (b) data collected with a request stored on the data aggregator, and (c) policy rules. The policy rules consist of integrity check rules on the platform and environment, a threshold value for the authentication score, and the alternate authentication method if the authentication score is less than a threshold value. After successful policy registration, the authentication consumer redirects the access request to the authentication engine along with request details. The authentication engine retrieves the request and queries the client

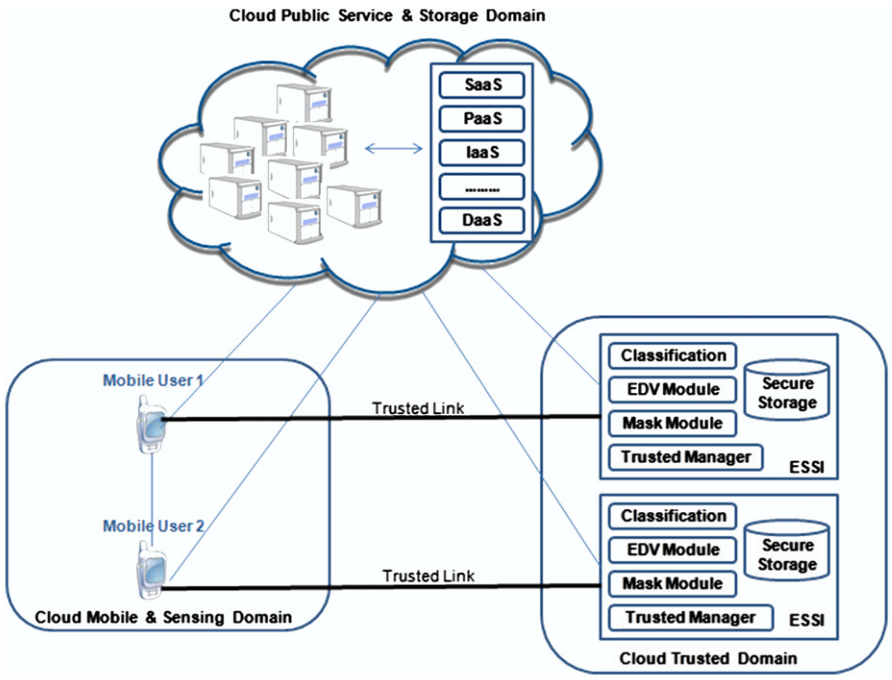

Fig. 14. Reference model for mobile cloud [63].

device or data collector for the required information to process the request. In return, the client device or data collector sends a generated report to the authentication engine. The authentication engine grants access permission to mobile clients through the authentication consumer, if authentication rules in the policy are satisfied.

The proposed scheme is dependent on the authentication service to identify a legitimate user. There is an option to host an authentication service on a third party. The growing number of users results in performance degradation when authentication service is hosted by third party. To avoid a performance degradation issue, multiple instances of authentication service may deploy within cloud on demand.

3.2.6. *Secure data processing framework for MCC*

The MobiCloud architecture proposed in [60] did not consider the privacy and security of users' data stored on cloud. Huang et al. [63] proposed a secure data processing model for MobiCloud that provides enhanced security and privacy protection for mobile users with the help of multi-tenant secure data management, trust management, and a *ESSI* data processing model. The proposed model consists of three domains: (a) cloud public service and storage domain, (b) cloud trusted domain, and (c) cloud mobile and sensing domain. The clone of each mobile device called *ESSI* is running under the supervision of the cloud trusted domain. The *ESSI* not only augments the processing and storage capabilities of the device but also provides enhanced security and privacy protection. Strict security policies are implemented in the cloud trusted domain through a distributed firewall system for the inspection of incoming and outgoing malicious packets to avoid data leakage. The detail of the model is shown in Fig. 14.

The authors assume that trusted authority is always available. The trusted authority controls the key distribution, certificate distribution, and user centric identity management. The trusted authority is responsible for deploying attribute based identity management [43] for each mobile device. The attribute based identity management scheme uses the publicly known identities of mobile users to generate the private keys for secure communication (discussed in Section 3.2.4). The identity can also be used as a signature to authenticate an individual. The proposed data processing framework introduces a secure and efficient mechanism to distribute the group key among group members to establish a one-to-many communication session in a dynamic communication environment, where a communication node may change anytime. The secure group communication involves an identity based signature scheme [44] for authentication, a threshold secret sharing scheme [60] to construct a data access policy, and an attribute based encryption scheme [46,48] for secure group key distribution and data access control. The presented multi tenant data management system divides data into two categories: (a) critical data and (b) normal data. Critical data is encrypted with a user generated key while normal data is encrypted with a cloud generated key. Incoming data flow received by the *ESSI* is classified as critical or normal data. If data is found to be critical, the Encryption Decryption and Verification (*EDV*) module is used to encrypt data and store in the secure storage of *ESSI* shown in Fig. 14. The normal data is stored on cloud through a masking procedure. The masking procedure removes the private information to preserve anonymity depending on user preference. The presented approach has a number of advantages over the traditional approaches, e.g. scalability, user critical data is not stored on cloud storage, computation is distributed between *ESSI* and a mobile device, and no single point of failure.

3.2.7. *A security framework of group location-based mobile applications in cloud computing*

Cloud offers an enormous amount of storage that motivates the application provider to outsource original database on cloud. One of the mobile applications that may take advantage of cloud storage is Location-Based Services (*LBS*) [64] for mobile. The *LBS* can utilize the cloud storage to maintain the large database that may contain historical place information, hotel information, or entertainment information depending on the application type. The enormous amount of storage enables a location-based service provider to store more data and strengthen the services. Chen and Wang [65] proposed a security framework for *LBS* that uses a location-based group scheduling service called *JOIN* [65] to investigate the security problem in *LBS*. The *JOIN* consumes cloud

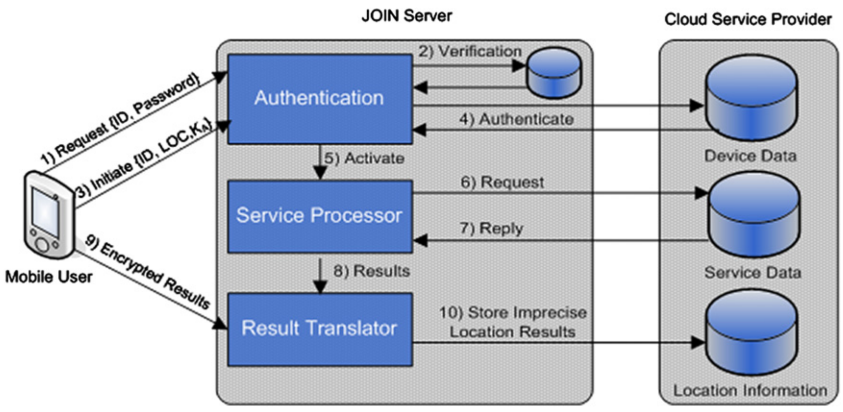

Fig. 15. Architecture of *JOIN* [65].

storage to store data on an outsourced database. The *JOIN* gathers information about friends around mobile user(s) and recommends some fascinating activity in the user(s) vicinity. There are three main components in *JOIN*: (a) users, (b) *JOIN* server, and (c) cloud database. The user is the mobile entity that utilizes the *JOIN* services. The *JOIN* server is responsible for authenticating users, facilitating the group scheduling activity, storing user data, and result translation. The cloud database stores data about devices, services, and location, while mobile user data is stored on the *JOIN* server. The architecture of *JOIN* is shown in Fig. 15.

Users have to register with the *JOIN* server to utilize services. For registration, the mobile user generates a key (K_A) by applying a hash function on International Mobile Subscribe Identity (*IMSI*).

$$K_A = H(IMSI). \tag{56}$$

The key information together with user identification, password, and group name is sent to *JOIN* server for registration. The *JOIN* server stores user information like identification, password, and group name in the database and generates key (K_B) using the hash function on user identification (*ID*) and K_A.

$$K_B = H(ID \parallel K_A). \tag{57}$$

The newly generated K_B and group name are delivered and stored on the cloud database for user authentication. To utilize the *JOIN* services, the mobile user has to login using *ID* and a password. After a successful login, the user may request *JOIN* services. The user can start an activity (invite friends to visit some place) by sending a K_A, group name, and location information to *JOIN* server. *JOIN* server regenerates K'_B on the basis of user *ID* and K_A as depicted in the following equation.

$$K'_B = H(ID \parallel K_A). \tag{58}$$

JOIN server compares newly computed K'_B with K_B stored in the cloud database. After successful verification, *JOIN* server sends a request to all other members of the same group. In return, each group member sends K_A, group name, and location information to *JOIN* server. *JOIN* server stores received information in a temporary table and verifies each member by comparing keys (K'_B and K_B). After successful authentication of group members, *JOIN* server generates the list of friends using a temporary table and list of famous places using a cloud database, near the mobile user vicinity. The generated information is encrypted using an Advanced Encryption Standard Algorithm (E) with initiator key (K_B).

$$C = E_{K_B}(Data). \tag{59}$$

The initiator receives the encrypted data (C) and generates the K_B using self *ID* and K_B as stated in Eq. (60). The encrypted file is decrypted (D) using K_B.

$$K_B = H(ID \parallel K_A) \tag{60}$$

$$Data = D_{K_B}(C). \tag{61}$$

The *JOIN* server keeps user's K_A and location information in the cloud database as a historic location record. Whenever any user wants to use the *JOIN* services, authentication is done with the help of K_A provided by the user and K_B stored on the cloud database. In *LBS*, the most important factor is to secure the identity of the mobile user. Data security is achieved by hiding the identity of the user from others by applying hash on *IMSI* to generate K_A. As the hash function is irreversible, the attacker cannot get the identity of mobile users.

3.2.8. *Privacy preserving schemes for scheduling services*

Bilogrevica et al. [66] proposed three privacy preserving solutions for mobile devices while using the scheduling services hosted on cloud. The proposed schemes utilize the homomorphic properties of well-known cryptographic systems to evaluate the common availability of the users securely. The main ideal on the proposed schemes is shown in Fig. 16.

The proposed algorithms find the common time slot j among a group of N users. The users are represented with u_i having private schedules x_i as expressed in the following equation.

$$x_i = b_{i,1}, b_{i,2}, \ldots, b_{i,m} \tag{62}$$

where $b_{i,j} = 1$, if the ith user is available in jth time slot, otherwise $b_{i,j} = 0$. Each user computes a private time schedule and applies transformation function (f) using the shared secret key of group so that an adversary cannot identify the user's schedule by observing the output.

$$f_i = f(b_{i,1}, b_{i,2}, \ldots, b_{i,m}). \tag{63}$$

The transformed information (f_i) is delivered to the scheduling server hosted on cloud. The scheduling server runs scheduling algorithm A on f_i to produce output ($f(Y)$).

$$f(Y) = A(f_1, f_2, f_3, \ldots, f_n). \tag{64}$$

The scheduling server sends $f(Y)$ to each user. The users evaluate common time slot (Y) by applying the transformation function (f^{-1}) on $f(Y)$. The Y contains the information about the availability of all users in each time slot represented with y^j. The value of y^j=*YES*, if all users are available in jth time slot, otherwise y^j=*NO*.

$$y^j = \begin{cases} YES, & \text{if } b_{i,j} = 1, \forall i \in n \\ NO, & \text{Otherwise.} \end{cases} \tag{65}$$

The proposed privacy preserving schemes utilize the homomorphic property of well-known cryptographic systems ElGamal [67], Paillier [68], and Goldwasser–Micali [69] for secure and efficient transformation of input and generation of output.

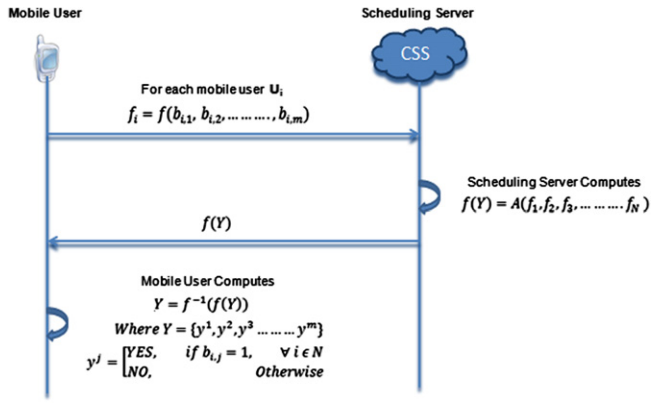
Mobile User
Scheduling Server
CSS
For each mobile user U_i
$f_i = f(b_{i,1}, b_{i,2}, \ldots\ldots, b_{i,m})$
Scheduling Server Computes
$f(Y) = A(f_1, f_2, f_3, \ldots\ldots f_N)$
$f(Y)$
Mobile User Computes
$Y = f^{-1}(f(Y))$
Where $Y = \{y^1, y^2, y^3 \ldots\ldots y^m\}$
$y^j = \begin{bmatrix} YES, & if\ b_{i,j} = 1, \quad \forall i \in N \\ NO, & Otherwise \end{bmatrix}$

Fig. 16. The proposed scheduling framework.

3.2.8.1. SchedElG algorithm. The scheme utilizes the homomorphic property of the *ELGamal* cryptographic system that allows the scheduling server to compute the aggregated availability of group members. The aggregated availability is evaluated by applying multiplication operations on an encrypted schedule of group members. The homomorphic property of *ELGamal* satisfies the following:

$$\begin{aligned} D\left(E_{K_p,r1}(M1) \cdot E_{K_p,r2}(M2)\right) &= D\left(\left(g^{r1}, M1 \cdot h^{r1}\right)\right. \\ &\quad \left.\cdot \left(g^{r2}, M1 \cdot h^{r2}\right)\right) = D\left(\left(g^{r}, (M1 \cdot M2)h^{r}\right) \cdot\right) \\ &= M1 \cdot M2 \end{aligned} \tag{66}$$

where $r = r_1 + r_2 \in \mathbb{Z}_q$ is selected randomly. The group members represent the availability $(b^*_{i,j})$ using the following equation.

$$b^*_{i,j} = \begin{bmatrix} 1, & \text{if } b_{i,j} = 1, \forall i \in n, \forall j \in m \\ R, & \text{if } b_{i,j} = 0 \end{bmatrix} \tag{67}$$

where $R \in \mathbb{Z}_q$ is selected randomly and a having value greater than one. The group members arrange the sequence of time slots according to permutation (p_j) and encrypt the scheduling bits as depicted in the Eq. (68).

$$E_i = \{E_{i,p_1}, E_{i,p_2}, E_{i,p_3}, \ldots, E_{i,p_m}\}, \quad \text{where } E_{i,p_j = }E_{p_k,r_{i,j}}(b^*_{i,j}). \tag{68}$$

The group members send encrypted scheduling information (E_i) to scheduling server. The scheduling server computes the aggregated availability of the group by applying multiplication operation on all group members' encrypted schedules shown below.

$$E_{schedule} = \prod_{i=1}^{N} E_{i,p_j}, \quad \forall j \in \{1, 2, 3, \ldots, m). \tag{69}$$

The scheduling server sends the $E_{schedule}$ value to group members. Each segment of the $E_{schedule}$ contains the product of the scheduled value of group members in common time slots encrypted with the group's common session key. The group members decrypt the message with common session key to obtain the aggregated availability (B^*_{pj}) in each time slot.

$$Y = D_{k_p}(E_{schedule}) \tag{70}$$

$$Y = \{B^*_{p_1}, B^*_{p_2}, B^*_{p_3}, B^*_{p_4}, \ldots, B^*_{p_m}\}. \tag{71}$$

The one value of B^*_{pj} ensures the availability of group members in the jth time slot. Any other value of B^*_{pj} confirms the unavailability of at least one group member in the jth time slot.

3.2.8.2. SchedPa algorithm. The scheme utilizes the homomorphic property of the *Paillier* cryptographic system that allows the scheduling server to compute the aggregated availability of all group members. The aggregated availability is evaluated by applying multiplication and exponential operations on encrypted schedule of the group members. The homomorphic property of *Paillier* satisfies the following:

$$\begin{aligned} &D\left(E_{K_p,r1}(M1) \cdot E_{K_p,r2}(M2) \mod n^2\right) \\ &\quad = M1 + M2 \mod n = D\left(E_{K_p,r1}(M1)^{M2} \mod n^2\right) \\ &\quad = M1 \cdot M2 \mod n \end{aligned} \tag{72}$$

where n is a composite number. The group members represent the availability $(b^-_{i,j})$ using the following equation.

$$b^-_{i,j} = \begin{bmatrix} 0, & \text{if } b_{i,j} = 1, \forall i \in n, \forall j \in m \\ r, & \text{if } b_{i,j} = 0 \end{bmatrix} \tag{73}$$

where $r \in \mathbb{Z}_n$ is randomly selected and having value greater than one. The group members arrange the sequence of time slots according to p_j and encrypt the scheduling bits shown below.

$$E_i = \{E_{i,p_1}, E_{i,p_2}, E_{i,p_3}, \ldots, E_{i,p_m}\}, \quad \text{Where } E_{i,p_j = }E_{p_k,r_{i,j}}(b^-_{i,j}). \tag{74}$$

The group members send calculated E_i to scheduling server. The scheduling server computes the aggregated availability of group by applying multiplication and exponential operations on group members' encrypted schedules represented in the following equation.

$$E_{schedule} = \left(\prod_{i=1}^{N} E_{i,p_j}\right)^{R}, \quad \forall j \in \{1, 2, 3, \ldots, m). \tag{75}$$

The scheduling server sends $E_{schedule}$ value to group members. Each segment of the $E_{schedule}$ contains the sum of the scheduled value of group members in common time slots encrypted with group's common session key. The group members decrypt the message with common session key to obtain the aggregated availability (B^-_{pj}) in each time slot.

$$Y = D_{k_p}(E_{schedule}) \tag{76}$$

$$Y = \{B^-_{p_1}, B^-_{p_2}, B^-_{p_3}, B^-_{p_4}, \ldots, B^-_{p_m}\}. \tag{77}$$

The zero value of B^-_{pj} ensures the availability of the all users in jth time slot. Any other value of B^-_{pj} confirms the unavailability of at least one group member in the jth time slot.

3.2.8.3. SchedGM algorithm. The scheme utilizes the homomorphic property ofthe *Goldwasser–Micali* cryptographic system that allows the scheduling server to compute the common availability of group members. The common availability is evaluated by applying the multiplication operations on the encrypted schedule of each group member. The homomorphic property of *Goldwasser–Micali* satisfies the following:

$$D\left(E_{K_p,r1}(M1) \cdot E_{K_p,r2}(M2) \mod n^2\right) = M1 \oplus M2. \tag{78}$$

The group members generate mask bits represented with S_i having the same length as schedule x_i and arrange the sequence of time slots according to p_j. Each group member masks the scheduling bits according to permutation shown below.

$$S_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,m}\}, \quad \text{where } c_{i,j} \in \{1, 0\} \tag{79}$$

$$b^{\oplus}_{i,\,p_j} = b_{i,p_j} \oplus c_{i,j}. \tag{80}$$

The group members are responsible for sending encrypted bit mask (E_i^C) and encrypted mask availability (E_i) to the scheduling server. The E_i^C and E_i are evaluated using the following equations.

$$E_i^C = \left[E_{k_p,r_{i,1}}\left(c_{i,1}\right), E_{k_p,r_{i,2}}\left(c_{i,2}\right), E_{k_p,r_{i,3}}\left(c_{i,3}\right), \ldots, E_{k_p,r_{i,m}}\left(c_{i,m}\right)\right] \quad (81)$$

$$E_i = \left[E_{i,p_1},\ E_{i,p_2}, E_{i,p_3}, \ldots, E_{i,p_m}\right]$$
$$\text{where } E_{i,p_j} = E_{k_p,r_{i,j}}(b_{i,p_j}^{\oplus}). \quad (82)$$

The scheduling server computes the availability of group by applying multiplication operations on E_{i,p_j} with the encrypted mask of all other group members.

$$E_{i,p_j}^{\oplus} = \left[E_{i,p_j} \times \prod_{k \neq i} E_{k_p}\left(c_{k,j}\right)\right],$$
$$\forall j \in \{1, 2, 3, \ldots, m\}, \forall i \in \{1, 2, 3, \ldots, N\}. \quad (83)$$

Afterwards, the scheduling server sends the masked availability information to group members. The group members decrypt the received message with a common session key and compare all masked individual schedules. The same value of the masked individual schedule for jth time slot confirms the availability of all group members in that slot.

The authors have implemented the scheduling schemes on a mobile device to evaluate the computation and communication performance. Experimental results and privacy analysis confirm that the proposed schemes provide a significant improvement in privacy, communication, and computation as compared to well-known scheduling solutions.

All the frameworks discussed under the category of application security frameworks have been presented in Table 2 chronologically. Each security framework has been evaluated with reference to evaluation criteria discussed in Section 2.2.

4. Discussion and future directions

4.1. Data security frameworks

In [38], Itani et al. proposed an energy efficient integrity verification scheme for mobile clients to verify the integrity of the files stored on a cloud server using an incremental message authentication code. The proposed scheme offloads most of the integrity verification jobs on a cloud service provider and *TTP* to minimize the processing overhead on the mobile client. The cloud service provider redirects the stored files towards the coprocessor when instructed by a mobile client. The coprocessor computes incremental *MAC* on received files for integrity verification. The proposed framework overlooked the privacy of uploaded files. Secondly, trusted coprocessors are installed on a cloud by *TTP* to ensure the integrity of mobile user's file/data. The coprocessor can handle a specific number of mobile users. The increase in the number of mobile users may result in performance degradation.

In [41], Jia et al. proposed a secure data service that outsources data and security management on cloud without disclosing any user information with the help of proxy re-encryption and identity based encryption schemes. Although the proposed secure data service has removed security management overhead from mobile users, still mobile users have to perform cryptographic operations before uploading a file on cloud. The cryptographic operations involve massive pairing evaluations and exponential calculations. The cryptographic operations consume a considerable amount of energy that needs be considered while designing a secure framework for *MCC*. Secondly, the cloud is responsible for performing the security management and re-encryption on behalf of the mobile user. The outsourcing of security management and re-encryption increase the utilization of cloud resources. The excessive use of cloud resources results in an overcharge to mobile user. There is a need for strong analysis that shows the tradeoff between resources consumption on cloud and energy consumption on a mobile device while using the secure data service.

In [45], Hsueh et al. proposed a scheme for smart phones that ensures the security, integrity, and authentication of mobile user data. The mobile user encrypts the files using traditional asymmetric encryption techniques. The encrypted files are stored on cloud servers along with mobile user name, signature, and password. The encrypted files along with user credentials may be stored on a cloud server hosted by an adversary. The adversary can utilize credentials to impersonate the user later on. Secondly, the proposed scheme ignored the processing and storage limitations of the device. The encryption, decryption, and even hash function applied on an entire file are performed on the mobile device.

In [46], Yang et al. proposed a public provable data possession scheme for a resource constrained mobile device that ensures privacy and confidentiality. *TTP* is responsible for handling encoding/decoding, encryption/decryption, signature generation, and verification on behalf of the mobile user. Although the offloading of mobile users' jobs on *TTP* saves energy, an increase in the number of mobile users results in performance degradation.

In [29], Ren et al. proposed three schemes to ensure the confidentiality and integrity of the user's files/data stored on fully distrusted cloud servers. The *EnS* uses standard cryptography functions to encrypt and decrypt a file on a mobile device. The standard encryption and decryption functions need processing power that consumes a considerable amount of energy on the device. The *CoS* overcomes the aforementioned deficiency by using a secrecy code on each part of share to achieve confidentiality. The *CoS* reduces processing overhead on the mobile device compared to *EnS*. To reduce more processing overhead on the device side, *ShS* was introduced. The *ShS* generates $(d-1)$ shares randomly. The dth share is produced with the help of two *xor* operations. The first *xor* operation is performed on $(d-1)$ randomly generated shares to produce *AR*. The second *xor* operation is performed on the original file and *AR* to obtain the dth share. All d shares are randomly stored on cloud servers. There is a possibility that all shares are stored on a single cloud server hosted by an adversary. The adversary can regenerate the original file by simply applying *xor* operations. The second possibility is that shares are stored on different cloud servers, even then a collusion attack is possible. The *ShS* requires less processing on a device but at the cost of compromised user privacy. The *EnS*, *CoS*, and *ShS* are not feasible for the sharing of files in a group. The user has to inform each group member individually about a file's password through phone, email, *SMS*, or any other medium. There should be some automated mechanism to share files or passwords among multiple users securely.

In [51], Zhou and Huang proposed a privacy preserving framework by extending the ciphertext policy attribute-based encryption scheme [52]. The proposed scheme offloads the processing and storage intensive encryption and decryption operations on cloud without revealing any information about data contents and security keys. The problem with a ciphertext policy attribute-based encryption scheme is that the ciphertext grows linearly with an increase in number of ciphertext attributes [70]. The increase in ciphertext involves more pairing evaluation and exponential operations while decrypting the ciphertext. As the proposed privacy preserving framework *PP-CP-ABE* is based on a ciphertext policy attribute-based encryption scheme, the same problem is inherited in *PP-CP-ABE*. There is a need for an attribute based encryption scheme that has a fixed ciphertext regardless of ciphertext attributes.

Table 2
Comparison of evaluated application security frameworks.

Frameworks	Application type	Security features (SF)			Assumptions (AP)		Scalability
		SF-1	SF-2	SF-3	AP-1	AP-2	
Zhang et al. [34]	Security framework for elastic mobile application model	Secure installation of elastic application	Authentication between weblets, secure migration	Authentication and authorization to access external web services	DEM on mobile device is trusted	CES on cloud is semi trusted	Highly scalable
Xiao and Gong [56]	Scheme for mobile cloud environment to generate the dynamic credential for mobile user	–	Authentication	–	Communication between ISP and cloud is secured	Communication within cloud is secured	Highly scalable
Wang and Wang [57]	Privacy preserving framework for mobile device while using *LBS*	Location privacy	–	–	Cloud service provider is semi trusted	Total number of mobile client in whole region is far greater than threshold *k*	Highly scalable
Huan et al. [60]	Framework (MobiCloud) to enhance the functionality of MANET and covers security aspects	Trust management	Risk management	Secure routing	–	–	Moderate
Chow et al. [62]	Policy based cloud authentication platform using implicit authentication	Security of mobile user's noticeable context and action stored on cloud	Authentication	–	–	–	Highly scalable
Huang et al. [63]	Secure data processing framework for MobiCloud	Security of data stored on cloud	Authentication	–	–	–	Highly scalable
Chen et al. [65]	Security framework for location-based grouped scheduling services	Identity privacy	Authentication	–	Cloud database is secured	Communication channel is secured	Highly scalable
Bilogrevica et al. [66] (*SchedElG*)	Privacy preserving solution for scheduling services on mobile device using homomorphic property of *ELGamal*	Privacy for scheduling operation	–	–	Mobile user is semi trusted	Third party server (cloud) is semi trusted	Highly scalable
Bilogrevica et al. [66] (*SchedPa*)	Privacy preserving solution for scheduling services on mobile device using homomorphic property of *Paillier*	Privacy for scheduling operation	–	–	Mobile user is semi trusted	Third party server (cloud) is semi trusted	Highly scalable
Bilogrevica et al. [66] (*SchedGM*)	Privacy preserving solution for scheduling services on mobile device using homomorphic property of *Goldwasser–Micali*	Privacy for scheduling operation	–	–	Mobile user is semi trusted	Third party server (cloud) is semi trusted	Highly scalable

4.2. Application security frameworks

In [34], Zhang et al. proposed an elastic mobile application model and covered the security aspects of the proposed model. The proposed security model deals with the secure installation of elastic application, authentication among weblets, secure migration of weblets, and authorization of weblets while using external web services. The proposed model safeguards the modification of a weblet only at installation time with the help of a developer signed hash value. When weblets are transferred to the cloud for execution, an attacker can perform an active man in the middle attack to insert malicious code in the weblet. The infected weblet may compromise *DEM* or *CES* to make changes in application configurations. The application configurations can be changed in a way such that computationally intensive and memory demanding jobs are executed on a mobile device to exhaust energy.

In [56], Xiao and Gong proposed a scheme to generate automatic dynamic credentials with the mutual coordination of mobile devices and cloud. The generated credentials are used to protect the mobile user from different type of attacks. The proposed scheme considers cloud as a fully trusted component to implement the solution accurately. The assumption for the proposed scheme is very strong which needs to be considered. Secondly, the mobile user has to update users' dynamic secrets frequently, resulting in a processing burden, communication delay, and loss of energy on a mobile device.

In [57], Wang and Wang proposed a top down spatial cloaking mechanism with and without optimization that utilizes cloud resources to provide a scalable, efficient, and improved privacy preserving framework for location-based services. The in-device spatial cloaking without optimization needs to communicate with the cloud server to get the number of users in different grid cells. The communication process induces delay and communication overhead as discussed in [57]. To overcome delay and communication overhead, a new technique was proposed called in-device spatial cloaking with optimization. The correctness of the new proposed technique is dependent on the historical lower bound of the number of users in each grid cell. The proposed technique predicts the number of users in each grid cell on the basis of the historical data that may be wrong in current scenario and results in user privacy loss. Secondly, the cloud is responsible for maintaining the historical data of each cell and sharing with mobile clients during the start up time. The maintenance and processing of historical data impose an extra burden on cloud as compared to former techniques.

In [62], Chow et al. proposed a policy based cloud authentication platform that addresses the client device authentication issue using implicit authentication. The noticeable user information is collected and stored on a data aggregator that may affect the user's privacy. To achieve privacy, mobile clients apply hash function on data (except *GPS* location) with a self-generated key and transfer the generated information to the data aggregator. An unusual pattern can be detectable without disclosing the noticeable user information. The mobile client has to apply a hash function on frequent generated information to achieve privacy. The frequent executions of a hash function on a mobile client demand processing and result in communication delay and energy loss.

In [60], Huan et al. proposed a new *MCC* framework that not only provides conventional computation services but also improves the functionality of *MANET* in terms of risk management, trust management, and secure routing. In spite of the advantages provided by the MobiCloud to *MANET*, the MobiCloud framework did not consider the trustworthiness of the cloud node. There should be a mechanism to store mobile user information on cloud servers in a secure manner.

In [63], Huang et al. proposed secure data processing model for MobiCloud that provides enhanced security and privacy protection for mobile users with the help of multi-tenant secure data management, trust management, and a *ESSI* data processing model. To provide strong security services to the user, the storage domain module and cloud trusted domain module are physically isolated. If the cloud trusted domain module is hosted by a trusted third party, there is an issue of scalability. The second option is to host the trusted domain module in another cloud service provider. The different cloud service providers may use a different set of *APIs* to implement cloud services. The interoperability between two different service providers may be an issue that should be considered before hosting the services.

In [65], Chen and Wang proposed a security framework for *LBS* and used a location-based group scheduling service called *JOIN* to investigate the security issues in *LBS*. If the *JOIN* server is hosted on cloud, the user's information (e.g. *ID*, *PWD*, and K_A) is visible to the cloud service provider. The authors did not introduce any mechanism to securely store the user credential information on the server. Secondly, the *JOIN* server uses the *AES* algorithm to encrypt the data that the mobile device has to decrypt. There is a need for a lightweight encryption technique that should be beneficial for mobile users while decrypting in terms of battery life and processing. Thirdly, the most important part of the proposed security mechanism is *IMSI*. If *IMSI* information is disclosed to an adversary, the security of the whole system fails.

In [66], Bilogrevica et al. proposed a privacy preserving solution for mobile devices while using the scheduling services hosted on cloud. The proposed schemes utilize the homomorphic properties of well-known cryptographic systems *ElGamal*, *Paillier*, and *Goldwasser–Micali* to evaluate the common availability of the mobile users securely. The mobile device is responsible for encrypting and encoding the messages to ensure indistinguishability [71]. Both operations demand processing that ultimately results in battery consumption on the device side. In proposed schemes, the mobile device encrypts and encodes a limited amount of data that does not significantly affect the performance of the system in terms of computation, delay, and energy loss.

5. Conclusion

The survey critically investigates different security frameworks proposed for the *MCC* environment. Most of the discussed security frameworks offload processor intensive jobs on cloud due to the resource limitation of mobile devices. Although the offloading increases the processing capability of the mobile device, the mobile user has to pay while using the cloud resources in a pay-as-you-go manner. Most of the discussed security frameworks overlooked the tradeoff between the energy consumption on the device and the expense of using cloud resources while designing a security framework.

A combination of cloud computing with virtualization provides efficient utilization of cloud resources with minimum cost. Despite the advantages provided by virtualization in *MCC*, new security threats need to be tackled due to the lack of perfect isolation among various virtual machine instances running on the same physical server.

The most challenging aspects in *MCC* are guaranteeing user privacy and the provision of mobile application security that uses cloud resources. To provide a secure *MCC* environment, service providers need to address issues pertaining to data security, network security, data locality, data integrity, web application security, data segregation, data access, authentication, authorization, data confidentiality, data breach issues, and various other factors. To achieve a secure *MCC* environment, security threats need to be studied and addressed accordingly.

Acknowledgment

This work is done under the Bright Spark Unit, University of Malaya, Malaysia. Samee U. Khan's work was partly supported by the Young International Scientist Fellowship of the Chinese Academy of Sciences, (Grant No. 2011Y2GA01).

References

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, Above the clouds: a Berkeley view of cloud computing, Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb. 2009.

[2] R. Ranjan, A. Harwood, R. Buyya, Grid federation: an economy based distributed resource management system for large-scale resource coupling, Technical Report GRIDS-TR-2004-10, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, 2004.

[3] R. Buyya, R. Ranjan, Federated resource management in grid and cloud computing systems, Future Generation Computer Systems 26 (8) (2006) 1189–1191.

[4] B. Sotomayor, R.S. Montero, I.M. Lorente, I. Foster, Virtual infrastructure management in private and hybrid clouds, IEEE Internet Computing 13 (5) (2009) 14–22.

[5] L. Wang, J. Tao, M. Kunze, H. Von, D. Kramer, W. Karl, Scientific cloud computing: early definition and experience, in: Proc. 10th IEEE Int. Conference on High Performance Computing and Communications, HPCC '08, Dalian, China, Sep. 2008.

[6] F. Hu, M. Qiu, J. Li, T. Grant, D. Tylor, S. Mccaleb, L. Butler, R. Hamner, A review on cloud computing: design challenges in architecture and security, Journal of Computing and Information Technology 19 (1) (2011) 25–55.

[7] A. Nathani, S. Chaudhary, G. Somani, Policy based resource allocation in IaaS cloud, Future Generation Computer Systems 26 (1) (2012) 94–103.

[8] L. Wang, G.V. Laszewski, A. Younge, X. He, M. Kunze, J. Tao, C. Fu, Cloud computing: a perspective study, New Generation Computing 28 (2) (2010) 137–146.

[9] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I.M. Lorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Benyhuda, W. Emmerich, F. Galan, The reservoir model and architecture for open federated cloud computing, IBM Journal of Research and Development 53 (4) (2009) 1–11.

[10] J. Murty, Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB, first ed., O'Reilly Media, 2008.

[11] E.Y. Chen, M. Itoh, Virtual smartphone over IP, in: Proc. IEEE Int. Symposium on A World of Wireless Mobile and Multimedia Networks, WoWMoM '10, Montreal, QC Canada, June 2010.

[12] Google App Engine, September 02, 2011. https://developers.google.com/appengine/.

[13] Force.com Apex Code Developer's Guide, September 06, 2011. http://www.salesforce.com/us/developer/docs/apexcode/index.htm.

[14] B.P. Rimal, E. Choi, I. Lumb, A taxonomy and survey of cloud computing systems, in: Proc. 5th Int. Joint Conference of INC, IMS and IDC, NCM '09, Seoul, Korea, Nov. 2009.

[15] H. Canepa, D. Lee, A virtual cloud computing provider for mobile devices, in: Proc. 1st ACM Workshop on Mobile Cloud Computing & Services Social Networks and Beyond, MCS '10, San Francisco, USA, June 2010.

[16] K. Kumar, Y.H. Lu, Cloud computing for mobile users: can offloading computation save energy? IEEE Journal Computer 43 (4) (2010) 51–56.

[17] D. Zissis, D. Lekkas, Addressing cloud computing security issues, Future Generation Computer Systems 28 (3) (2012) 583–592.

[18] S. Subashini, V. Kavitha, A survey on security issues in service delivery models of cloud computing, Journal of Network and Computer Applications 34 (1) (2011) 1–11.

[19] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems 25 (6) (2009) 599–616.

[20] C.O. Diaz, M. Guzek, J.E. Pecero, P. Bouvry, S.U. Khan, Scalable and energy-efficient scheduling techniques for large-scale systems, in: Proc. 11th IEEE Int. Conference on Computer and Information Technology, CIT '11, Pafos, Cyprus, Sep. 2011.

[21] C. Cai, L. Wang, S.U. Khan, J. Tao, Energy-aware high performance computing: a taxonomy study, in: Proc. 17th IEEE Int. Conference on Parallel and Distributed Systems, ICPADS '11, Tainan, Taiwan, Dec. 2011.

[22] I. Goiri, J.L. Berral, J.O. Fitó, F. Julià, R. Nou, J. Guitart, R. Gavaldà, J. Torres, Energy-efficient and multifaceted resource management for profit-driven virtualized data centers, Future Generation Computer Systems 28 (5) (2012) 718–731.

[23] D. Kliazovich, P. Bouvry, S.U. Khan, DENS: data center energy-efficient network-aware scheduling, in: Proc. ACM/IEEE Int. Conference on Green Computing and Communications, GreenCom '10, Hangzhou, China, Dec. 2010.

[24] P. Lindberg, J. Leingang, D. Lysaker, S.U. Khan, J. Li, Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems, Journal of Supercomputing 59 (1) (2012) 323–360.

[25] D.M. Quan, F. Mezza, D. Sannenli, R. Giafreda, *T*-alloc: a practical energy efficient resource allocation algorithm for traditional data centers, Future Generation Computer Systems 28 (5) (2012) 791–800.

[26] C.O. Diaz, M. Guzek, J.E. Pecero, G. Danoy, P. Bouvry, S.U. Khan, Energy-aware fast scheduling heuristics in heterogeneous computing systems, in: Proc. ACM/IEEE/IFIP Int. Conference on High Performance Computing and Simulation, HPCS '11, Istanbul, Turkey, July 2011.

[27] S.U. Khan, C. Ardil, A weighted sum technique for the joint optimization of performance and power consumption in data centers, International Journal of Electrical, Computer, and Systems Engineering 3 (1) (2009) 35–40.

[28] S.U. Khan, A self-adaptive weighted sum technique for the joint optimization of performance and power consumption in data centers, in: Proc. 22nd Int. Conference on Parallel and Distributed Computing and Communication Systems, PDCCS '09, Louisville, KY, USA, Sep 2009.

[29] W. Ren, L. Yu, R. Gao, F. Xiong, Lightweight and compromise resilient storage outsourcing with distributed secure accessibility in mobile cloud computing, Journal of Tsinghua Science and Technology 16 (5) (2011) 520–528.

[30] D. Kovachev, Y. Cao, R. Klamma, Mobile cloud computing: a comparison of application models, Computing Research Repository, CoRR, vol. abs/1009.3088, 2010.

[31] L. Guan, X. Ke, M. Song, J. Song, A survey of research on mobile cloud computing, in: Proc. 10th IEEE/ACIS International Conference on Computer and Information Science, ICIS '11, Sanya, China, Nov. 2011.

[32] H.T. Dinh, C. Lee, D. Niyato, P. Wang, A survey of mobile cloud computing: architecture, applications, and approaches, Wireless Communication and Mobile Computing. http://dx.doi.org/10.1002/wcm.1203 (in press).

[33] N. Santos, K.P. Gummadi, R. Rodrigues, Towards trusted cloud computing, in: Proc. Hot Topics in Cloud Computing, HotCloud '09, San Diego, CA, June 2009.

[34] X. Zhang, J. Schiffman, S. Gibbs, A. Kunjithapatham, S. Jeong, Securing elastic applications on mobile devices for cloud computing, in: Proc. ACM workshop on Cloud computing security, CCSW '09, Chicago, IL, USA, Nov. 2009.

[35] B.G. Chun, P. Maniatis, Augmented smartphone applications through clone cloud execution, in: Proc. 12th Workshop on Hot Topics in Operating Systems, HotOS '09, Monte Verità, Switzerland, May 2009.

[36] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, G. Alonso, Calling the cloud: enabling mobile phones as interfaces to cloud applications, in: Proc. 10th ACM/IFIP/USENIX Int. Conference on Middleware, Middleware '09, Champaign, IL, USA, Nov. 2009.

[37] J. Rellermeyer, O. Riva, G. Alonso, Alfredo: an architecture for flexible interaction with electronic devices, in: Proc. 9th ACM/IFIP/USENIX Int. Conference on Middleware, Middleware '08, Leuven, Belgium, Dec. 2008.

[38] W. Itani, A. Kayssi, A. Chehab, Energy-efficient incremental integrity for securing storage in mobile cloud computing, in: Proc. Int. Conference on Energy Aware Computing, ICEAC '10, Cairo, Egypt, Dec. 2010.

[39] M. Bellare, O. Goldreicha, S. Goldwasser, Incremental cryptography: the case of hashing and signing, in: Proc. 14th Annual Int. Cryptology Conference on Advances in Cryptology, Santa Barbara, California, USA, Aug. 1994.

[40] M. Bellare, O. Goldreich, S. Goldwasser, Incremental cryptography and application to virus protection, in: Proc. 27th Annual ACM Symposium on Theory of Computing, STOC '95, Las Vegas, NV, USA, May 1995.

[41] W. Jia, H. Zhu, Z. Cao, L. Wei, X. Lin, SDSM: a secure data service mechanism in mobile cloud computing, in: Proc. IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, Shanghai, China, Apr. 2011.

[42] J. Shao, Z. Cao, CCA-secure proxy re-encryption without pairings in public key cryptography, in: Proc. 12th Int. Conference on Practice and Theory in Public Key Cryptography, PKC '09, Irvine, CA, USA, Mar. 2009.

[43] S. Yu, C. Wang, K. Ren, W. Lou, Achieving secure scalable and fine-grained data access control in cloud computing, in: Proc. IEEE INFOCOM, INFOCOM '10, San Diego, CA,USA, Mar. 2010.

[44] M. Green, G. Ateniese, Identity-based proxy re-encryption, in: Proc. 5th Int. Conference on Applied Cryptography and Network Security, ACNS '07, Zhuhai, China, June 2007.

[45] S.C. Hsueh, J.Y. Lin, M.Y. Lin, Secure cloud storage for conventional data archive of smart phones, in: Proc. 15th IEEE Int. Symposium on Consumer Electronics, ISCE '11, Singapore, June 2011.

[46] J. Yang, H. Wang, J. Wang, C. Tan, D. Yu1, Provable data possession of resource constrained mobile devices in cloud computing, Journal of Networks 6 (7) (2011) 1033–1040.

[47] Q. Wang, C. Wang, J. Li, K. Ren, W. Lou, Enabling public verifiability and data dynamics for storage security in cloud computing, in: Proc. 14th European Conference on Research in Computer Security, ESORICS '09, Saint Malo, France, Sep. 2009.

[48] D.A. Carts, A review of the Diffie–Hellman algorithm and its use in secure internet protocols, November 02, 2011. http://www.sans.org/reading_room/whitepapers/vpns/review-diffie-hellman-algorithm-secure-internet-protocols_751.

[49] D. Boneh, C. Gentry, Aggregate and verifiably encrypted signatures from bilinear maps, in: Proc. 22nd Int. Conference on Theory and Applications of Cryptographic Techniques, EUROCRYPT '03, Warsaw, Poland, May 2003.

[50] R.C. Merkle, Protocols for public key cryptosystems, in: Proc. IEEE Symposium on Security and Privacy, Oakland, California, USA, Apr. 1980.

[51] Z. Zhou, D. Huang, Efficient and secure data storage operations for mobile cloud computing, IACR Cryptology ePrint Archive: 185, 2011.

[52] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: Proc. 28th IEEE Symposium on Security and Privacy, SP '07, California, USA, May 2007.

[53] A. Shamir, How to share a secret, Communications of the ACM 22 (11) (1979) 612–616.

[54] Z. Zhou, D. Huang, An optimal key distribution scheme for secure multicast group communication, in: Proc. 29th Conference on Information Communications, INFOCOM '10, San Diego, USA, Mar. 2010.
[55] X. Zhang, S. Jeong, A. Kunjithapatham, S. Gibbs, Towards an elastic application model for augmenting computing capabilities of mobile platforms, Journal of Mobile Networks and Applications 16 (3) (2011) 270–284.
[56] S. Xiao, W. Gong, Mobility can help: protect user identity with dynamic credential, in: Proc. 11th Int. Conference on Mobile Data Management, MDM '10, Missouri, USA, May 2010.
[57] S. Wang, X.S. Wang, In-device spatial cloaking for mobile user privacy assisted by the cloud, in: Proc. 11th Int. Conference on Mobile Data Management, MDM '10, Missouri, USA, May 2010.
[58] W.G. Aref, H. Samet, Efficient processing of window queries in the pyramid data structure, in: Proc. 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS '90, Nashville, TN, USA, April 1990.
[59] M.F. Mokbel, C. Chow, W.G. Aref, The new casper: query processing for location services without compromising privacy, in: Proc. 32nd Int. Conference on Very Large Databases, VLDB '06, Seoul, Korea, Sep. 2006.
[60] D. Huan, X. Zhang, M. Kang, J. Luo, MobiCloud: building secure cloud framework for mobile computing and communication, in: Proc. 5th IEEE Int. Symposium on Service Oriented System Engineering, SOSE '10, Nanjing, China, June 2010.
[61] J. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, J. Luo, NetFPGa-an open platform for gigabitrate network switching and routing, in: Proc. IEEE Int. Conference on Microelectronic Systems Education, MSE '07, San Diego, California, June 2007.
[62] R. Chow, M. Jakobsson, R. Masuoka, J. Molina, Y. Niu, E. Shi, Z. Song, Authentication in the clouds: a framework and its application to mobile users, in: Proc. ACM Cloud Computing Security Workshop, CCSW '10, Chicago, USA, Oct. 2010.
[63] D. Huang, Z. Zhou, L. Xu, T. Xing, Y. Zhong, Secure data processing framework for mobilecloud computing, in: Proc. IEEE INFOCOM Workshop on Cloud Computing, INFOCOM '11, Shanghai, China, June 2011.
[64] L. Barkhuus, A. Dey, Location-based services for mobile telephony: a study of user's privacy concerns, in: Proc. Int. Conference on Human–Computer Interaction, INTERACT '03, Zurich, Switzerland, Sep. 2003.
[65] Y.J. Chen, L.C. Wang, A security framework of group location-based mobile applications in cloud computing, in: Proc. Int. Conference on Parallel Processing Workshops, ICPPW '11, Taipei, Taiwan, Sep. 2011.
[66] I. Bilogrevica, M. Jadliwalaa, P. Kumarb, S.S. Waliab, J.P. Hubauxa, I. Aadc, V. Niemic, Meetings through the cloud: privacy-preserving scheduling on mobile devices, Journal of Systems and Software 84 (11) (2011) 1910–1927. (Special Issue on Mobile Applications: Status and Trends).
[67] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Transactions on Information Theory 31 (4) (1985) 469–472.
[68] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: Proc. 7th Int. Conference on Theory and Application of Cryptographic Techniques, EUROCRYPT '99, Prague, Czech Republic, May 1999.
[69] S. Goldwasser, S. Micali, Probabilistic encryption, Journal of Computer and System Sciences 28 (2) (1984) 270–299.
[70] K. Emura, A. Miyaji, A. Nomura, K. Omote, M. Soshi, A ciphertext-policy attribute-based encryption scheme with constant ciphertext length, in: Proc. 5th Int. Conference on Information Security Practice and Experience, ISPEC '09, Xi'an, China, Apr. 2009.
[71] G. Castagnos, B.C. Mames, Towards a DL-based additively homomorphic encryption scheme, in: Proc. 10th Int. Information Security Conference, ISC '07,Valparaíso, Chile, Oct. 2007.

Abdul Nasir Khan received the M.C.S. and M.S. (CS) degrees from the COMSATS Institute of Information Technology, Abbottabad in 2005 and 2008 respectively. Currently, he is pursuing his Ph.D from the University of Malaya, Malaysia. He has been serving as a Lecturer in the Department of Computer Science, COMSATS Institute of Information Technology for three years. He has published 2 international conference and 2 international journal papers. His research interests are in various aspects of network security and distributed computing.

M.L. Mat Kiah received her B.Sc. (Hons) in Computer Science from University of Malaya (UM), Malaysia in 1997, M.Sc. in 1998 and Ph.D. in 2007 from Royal Holloway, University of London, UK. She joined the Faculty of Computer Science & Information Technology, UM as a tutor in 1997. Her current research interests include key management, secure group communication and wireless mobile security. She is also interested in routing protocols and mobile Ad-Hoc networks. A total of 28 (journal: 16, conference: 11, book chapter: 01) publications are attributed to her name.

Samee U. Khan is Assistant Professor of Electrical and Computer Engineering at the North Dakota State University, Fargo, ND, USA. Prof. Khan has extensively worked on the general topic of resource allocation in autonomous heterogeneous distributed computing systems. As of late, he has been actively conducting cutting edge research on energy-efficient computations and communications. A total of 117 (journal: 42, conference: 54, book chapter: 12, editorial: 6, technical report: 3) publications are attributed to his name. For more information, please visit: http://sameekhan.org/.

Sajjad A. Madani is an Associate Professor in the COMSATS Institute of Information Technology, Abbottabad. Previous to that, he was with the Institute of Computer Technology from 2005 to 2008 as a Guest Researcher where he did his Ph.D. research. Prior to joining ICT, he taught at the COMSATS Institute of Information Technology for a period of two years. He carried out an M.S. in Computer Sciences from Lahore University of Management Sciences (LUMS), Pakistan with excellent academic standing. His areas of interest include low power wireless sensor networks and the application of industrial informatics to electrical energy networks. He has published more the 30 papers in international conferences and journals.