Secure Mobile Cloud Computing and Security Issues

Qijun Gu and Mina Guirguis

Abstract The proliferation of mobile devices, coupled by the increase in their capabilities, have enabled the establishment of a rich mobile computing platform that can be utilized in conjunction with cloud services. In this chapter, we overview the latest mobile computing models and architectures focusing on their security properties. In particular, we study a wide range of threats against the availability, privacy and integrity of mobile cloud computing architectures in which the mobile devices and the cloud jointly perform computation. We then present defense mechanisms that ensure the security of mobile cloud computing architectures and their applications. Throughout the chapter, we identify potential threats as well as possible opportunities for defenses.

1 Introduction

Cloud computing is emerging as a revolutionary technology that is transforming the way we sense, compute and communicate into a new era. Meanwhile, mobile devices (e.g., smartphones) are becoming more advanced and pervasive, providing an information processing platform for mobile users. Due to recent technological advances in hardware, new capabilities in mobile devices have enabled the support of a wide range of applications. Despite such advances, however, mobile devices are still limited when compared to traditional computers and cannot effectively execute compute-intensive applications. As a result, mobile cloud computing emerged as a promising solution that conjugates the advantages of mobile devices with traditional cloud computing. In a mobile cloud computing environment, mobile devices offload intensive computational jobs, that they cannot perform locally, to the cloud.

Q. Gu (✉) • M. Guirguis
Texas State University-San Marcos, San Marcos, TX, USA
e-mail: qijun@txstate.edu; msg@txstate.edu

K.J. Han et al. (eds.), *High Performance Cloud Auditing and Applications*,
DOI 10.1007/978-1-4614-3296-8_3,

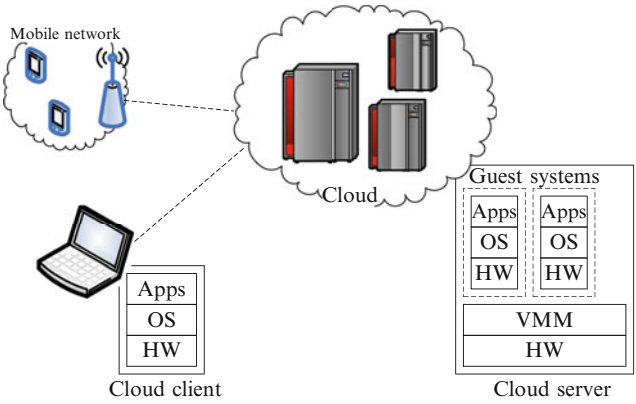

Fig. 1 Mobile computing cloud architecture

The coordination is conducted in a seamless manner so that users can experience and utilize a richer computational platform beyond the capabilities of the mobile devices.

There are many scenarios in which computation can be performed effectively on a mobile cloud computing platform. Consider the following two scenarios. The first one is when intensive computation is required. An example would be running virus scanning and application validation tools, which require significant portions of CPU time and memory on the mobile devices. A solution is to offload the execution of such tools to the cloud and let the devices perform the imminent jobs [25]. In the cloud, multiple virtualized phones can be built to scan applications for viruses and monitor running ones for validation. Another example is image stitching [62], whereby mobile users can take pictures of a scene and upload them to the cloud. Then, the cloud can realign the pictures and stitch them to produce a full-scale, detailed, and up-to-date image of the scene. The second scenario is when no feasible access to traditional cloud services is available. Consider the example outlined in [56], where a child is lost in a parade. Tourists can upload recently taken pictures to computers in a nearby police station. In the police station, image recognition software can run on the pictures to detect the location in which the child was last seen. In this example, tourists do not need to access any online cloud service to help finding the lost child.

A typical cloud system with mobile devices is illustrated in Fig. 1. In this system, mobile devices function as clients in a similar way to other regular computers. They offload computing jobs to cloud servers. In the cloud, the servers build virtualized guest systems that run mobile applications on virtual hardware and virtual operating systems. The guest systems are managed by the virtual machine manager (VMM) that bridges the host system with the guest systems. The VMM virtualizes, partitions, and allocates computing resources of the host system to the guest systems to provide computing services. In this chapter, we will discuss other proposed architectures for mobile cloud computing that support the needs of new applications.

Mobile cloud computing brings in a set of new challenges, especially when it comes to the security of the applicants, the availability of services and the privacy of the users. Due to the inherent complexity in mobile cloud computing architectures, attackers can target and exploit a much wider range of resources/protocols when compared to traditional client-server architectures. For example, on the mobile device, attackers can exploit vulnerabilities in the *mobile applications (Apps)*, the mobile operating systems and access to the mobile network. On the cloud, attackers can exploit vulnerabilities in cloud management, cloud virtualization and cloud access protocols. Recent attacks have shown that personal information can be disclosed, computing tasks can be maliciously altered and cloud services can be disabled for mobile users. Thus, it is critical to examine the security issues involving mobile cloud computing. In this chapter, our main goal is to identify threats against mobile cloud computing architectures and inspect the applicability of recent security solutions to address such threats.

This chapter is organized as follows. We overview a wide spectrum of mobile cloud computing architectures in Section 2. At one end of the spectrum, we have architectures that rely completely on mobile devices performing the computation, while at the other end of the spectrum, we have architectures that offload computation to the cloud. In Section 3, we examine security threats in mobile cloud computing along three security properties: availability, privacy, and integrity. We will examine how and what threats can compromise the security properties in mobile cloud computing architectures. In Section 4, we summarize defense mechanisms that ensure the security of mobile computing architectures against the outlined threats. We will first look into the applicable defenses to various components in the cloud and then summarize the security solutions designed specially for mobile cloud computing. Finally, we will conclude this chapter in Section 5.

2 Overview of Mobile Computing in the Cloud

In this section, we first review three newly proposed and representative cloud architectures for mobile computing. These architectures are designed to utilize the new features of mobile devices and satisfy new needs of mobile applications. Then, we discuss the performance and security issues of mobile computing in the cloud.

2.1 Mobile Client-Server Architectures

Mobile devices, such as smartphones, have enabled rich user experiences with Internet access, Global Positioning System (GPS), sensors and various applications. They naturally became an entry point for an exploding number of mobile users to cloud services to perform computing-intensive applications that are beyond the capabilities of mobile phones. A straightforward idea to support mobile users

with cloud computing is to apply the classical client-server model for mobile applications. Within this model, mobile phones function as thin clients in the sense that they are similar to an ordinary client computer that only provides a user interface (UI) to browse a server, and cloud servers run all applications. The mobile phone requests the execution of computational intensive applications from the cloud in a manner similar to requesting resources from web servers. However, this traditional client-server model does not take any advantage of the smartphones and overlooks many of the new and unique features they possess.

One main feature that draws smartphones apart from traditional mobile phones and desktop computers is their context awareness with regards to mobile computing. The context awareness is enabled by rich sensors on the smartphones. In particular, the context includes three components: spatial context, activity context, and group context [10]. The spatial context is harvested by the phone's GPS sensor and location-based service (LBS). It provides current location and past mobility traces to mobile applications. The activity context represents the user's activities, for example, whether he is driving, shopping, or talking. The activity information enables customized mobile applications to provide needed functionality to the user. The group context represents the surrounding mobile devices and users. It is the collection of their spatial context and activity context that enable mobile applications to interact with other nearby mobile devices. Thus, the context provides extra information to service providers so that they can perform computing according to the needs of mobile users given their context.

Another feature that makes mobile cloud computing different from traditional client server computing model is that mobile users dictate the computation on the cloud servers. Even though mobile applications can be executed in cloud servers, it is the mobile devices that decide how to execute mobile applications. When a mobile device runs an application, it identifies the computing-intensive portion of the application, and offloads that portion to the cloud server for processing. The cloud servers are typically considered as secondary processors and storage that provide extra computing resources for performing mobile applications. The coordination of such offloaded computing is mainly controlled by the mobile devices.

To meet the new requirements of mobile computing and exploit the new features of mobile devices, two types of new client-server architectures were proposed for augmenting mobile applications in the cloud. One architecture is based on a whole image clone [12], which is illustrated in Fig. 2a. A clone of a mobile phone is created in the cloud. The state of the phone and the clone is synchronized periodically or on-demand. During execution of an application, if the phone detects a block of computation that needs to be executed in the cloud, the application process on the phone enters a sleep state. The process is then continued in the cloud. When the cloud completes the execution, it updates the state of the phone. Then, the phone resumes the application process.

Rather than cloning the complete image of a phone, another architecture modularizes mobile applications and inserts a middleware layer between the applications and the operating systems of the phone [26]. The architecture is illustrated in Fig. 2b. A mobile application is partitioned into multiple modules according to

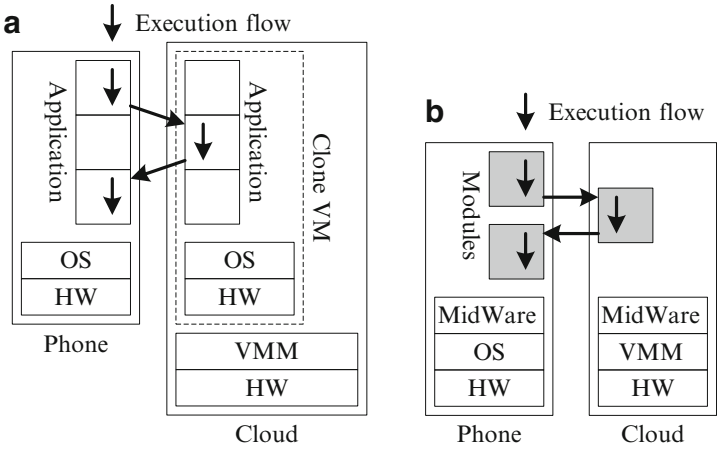

Fig. 2 Mobile cloud client-server architectures. (**a**) Whole image clone. (**b**) Code partitioning

their functionalities. Data and functionality dependencies are added as edges among modules. The modules are then labeled as movable or non-movable based on their resource requests, such as computation needs, storage needs, and amounts of interactions. Because the movable modules can migrate between the mobile devices and the cloud, the mobile applications are called *elastic applications*. Static and dynamic partition algorithms were developed to optimize the performance of offloading [13, 14, 26].

The movable modules are offloaded to the cloud when the mobile devices cannot execute them with their resources. The middleware, namely *elastic manager*, handles the migration of data and code associated with the offloaded modules. The elastic manager can be built on top of a web service [10], and correspondingly the application modules become weblets. The mobile devices request the migration and execution of the weblets through extended web requests. The elasticity manager can also be built as Java virtual machine (JVM) based modular framework [26], and the application modules are objects that can be loaded and executed as a Java class in this framework.

2.2 Cloudlet Architectures

Although the client-server cloud architecture can support the needs of mobile computing, it is affected by the inherent limitation of the access links of mobile devices. The fundamental limitation is the latency in the network. In the cloud, mobile devices interact with cloud servers over wireless wide area network (WWAN) and the Internet. Both networks incur non-negligible round-trip delay and jitter for interactive mobile applications. Such delay and jitter largely result in negative impact on the user's perception and cognitive engagement. Another latency

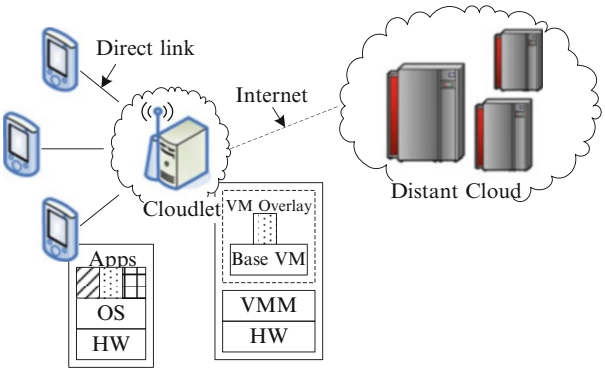

Fig. 3 Cloudlet architecture

is incurred in bulk transfers of data and code over bandwidth-limited WWAN. Even though wireless broadband technologies have been improving greatly, the bandwidth of WWAN is still significantly smaller than the bandwidth of wireless local area network (WLAN). The transfer time of bulk data is still far from satisfactory in WWAN.

To overcome the latency limitation with powerful but distant cloud servers, a new cloudlet architecture is proposed in [57], in which a resource-rich cloudlet is deployed close to the mobile users. The architecture is illustrated in Fig. 3.

A cloudlet is composed of powerful computers or a cluster that has sufficient computing capability and power. It provides cloud service to the mobile devices over a one-hop high-bandwidth wireless access such as WiFi. The advantages of using cloudlets arise from the real-time interactive response due to short and bounded-latency between the mobile devices and the cloudlets. A cloudlet can be integrated with a wireless access point (AP) for ease of management.

Utilizing cloudlets requires new mechanisms rather than simply treating cloudlets as traditional cloud servers so that the short latency of the cloudlets can be fully exploited. The authors in [57] proposed a transient cloudlet solution, namely dynamic Virtual Machines (VM) synthesis. When a mobile device needs a cloudlet to execute an application, the mobile device produces a *VM overlay* for the application. The VM overlay contains the soft state of the execution of the application. Then, the device delivers the VM overlay to the cloudlet. The cloudlet is pre-installed with a base VM that has the minimum system core components but can support running most applications. The cloudlet merges the overlay with the base and then executes the application from the state stored in the VM overlay. Once the job is done, the cloudlet returns the results to the mobile device and discards the VM overlay.

To verify the feasibility and performance of the cloudlet architecture, the proof-of-concept prototype Kimberley was implemented in VirtualBox. The prototype created VM overlays for a collection of Linux applications, such as AbiWord, GIMP and PathFind. The size of the compressed VM overlays ranges from 100

to 200 MBs in comparison with the full Linux VM image size of several gigabytes. Furthermore, the experiments with the prototype showed that the processing time of a VM overlay mainly includes the time of applying VM overlay, decompressing VM overlay, transferring VM overlay, and compressing private data. The reduction of the total time will lead to a more practical cloudlet architecture.

2.3 Ad Hoc Mobile Cloud Architectures

Although the first two cloud architectures for mobile computing are promising in managing and executing applications for mobile devices, they need an infrastructure to host the cloud service and to provide access to the mobile devices. It is not unusual that the access to such infrastructure may be unavailable to mobile users under various circumstances. For example, during a natural disaster (e.g., hurricane) that destroys part of the infrastructure, mobile devices may hardly access any services other than detecting themselves over Bluetooth and WiFi in an ad hoc mode. It may also be too expensive to access cloud service through the infrastructure, especially in areas where only high cost wireless data links (e.g., cellular link and satellite link) are available. Considering that the volume of bulk data and code to be offloaded is much higher than the amount of voice data, the cost of performing cloud computing via these high cost wireless links may largely negate the advantages obtained from the cloud services.

A new ad hoc mobile cloud solution was proposed in [31] to address the problems raised when no access to the cloud services is available. The solution is specially suitable to the needs of mobile users who are in close locations and share common activities. For example, when a tourist is visiting museums, he may find interests in the description of exhibits. He can take pictures of the text and run text-recognition software for recognition so that he can store the text on his mobile phone. However, processing the whole text requires more computing resources than his phone capabilities. He then looks for help from nearby tourists. Because other tourists may be interested in the description as well, they form an ad hoc network using the low cost WiFi communication and perform the text-recognition. The recognized text is then stored on their phones.

The adhoc mobile cloud architecture is illustrated in Fig. 4 to support this kind of location-bounded group activities [31]. The architecture consists of four main components for managing resources, applications, contexts, and offloading to the cloud. The *resource manager* profiles the needs of the applications and monitors the available resource. These profiles along with current resource information are checked upon the execution of an application to decide if the support from the cloud is needed. The *application manager* handles loading and executing applications. If the cloud is needed for launching an application, the manager adds extra information to the application for offloading. The *context manager* monitors the location and the number of mobile devices in the vicinity. This context information provides mobility traces of devices and builds social relations among them. The *offloading manager*

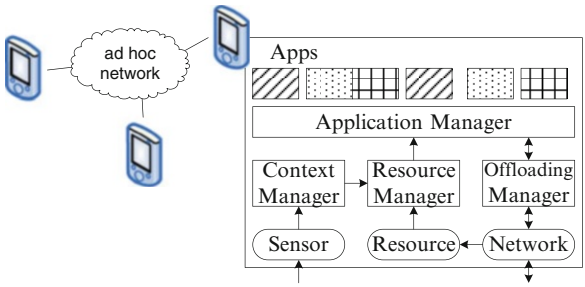

Fig. 4 Ad hoc mobile cloud architecture

partitions, sends and manages jobs from the mobile devices to other devices. It also processes and manages jobs received from other devices.

A concern of this ad hoc mobile cloud architecture is that the overall performance of the offloaded applications will be worse than running them on a single device because of the offloading overhead. To address this concern, a proof-of-concept implementation was built in JamVM and tested with Hadoop. The results showed that the execution with offloading was only 1% slower than regular execution. The execution time was further split to two parts: 44% for offloading preparation and waiting and 56% for processing and synthesizing.

2.4 Performance and Security Issues

In the following part, we summarize the performance and security issues in recent research in mobile cloud computing and discuss their interrelation.

2.4.1 Performance

The performance of mobile computing in the cloud is often considered along two dimensions: (1) the resources needed to support mobile computing in the cloud and (2) the overhead incurred for processing mobile applications, in both the mobile devices and the cloud.

Because mobile applications are offloaded to the cloud, the cloud needs to allocate resources to accommodate the offloaded portions of the mobile applications. The resources in the cloud are usually measured as (1) the size of memory where code and data are stored, (2) the size of the offloaded portions of the mobile application that are transferred, (3) the bandwidth needed for transferring offloaded applications, (4) the number of CPU cores or time slices for computing, among others.

The process of offloading and execution of mobile applications in the cloud is carried out through several procedures, such as partitioning, migration and execution. The overhead is often measured as processing times, which include

(1) the time for profiling and detecting the resources needed for running the applications in the mobile devices, (2) the time spent in partitioning and generating the offloaded portions of applications, (3) the time spent transferring the offloaded portions and the results between the mobile devices and the cloud, (4) the time spent merging and/or installing the offloaded portions into the guest systems in the cloud, (5) the time spent executing the offloaded portions in the cloud, (6) the interactive latency and jitter between the mobile devices and the cloud, and so on. Another unique measurement of the overhead in mobile computing is the power consumption on the mobile devices. It is as important as other performance factors in determining the feasibility of mobile computing in the cloud.

2.4.2 Security

The security of mobile cloud computing [2, 3, 7, 19, 45] covers a vast range of aspects and affects all components in the cloud. The security issues include the confidentiality and privacy of users' personal data, the integrity of the computation, the availability of the cloud services, the risk analysis of cloud configurations, the auditing of computation and information management, the accounting of the allocations of the resources, the secure management of the guest systems, the authentication of the users and devices, and key management with running applications, just to name a few.

Among all these issues, we will focus on the integrity, confidentiality and availability issues that are closely related to computing. The integrity issue arises since the mobile devices are less trustable and the mobile applications are offloaded to the cloud. The computational results from untrusted devices and offloaded applications need to be authenticated and verified. The confidentiality issue is due to sharing the code and data in cloud settings. Critical information, such as personal data, in mobile devices may be accessible from the cloud as the information is also backed up in the cloud. The availability issue is inherent to mobile computing due to the limited resources (e.g., cellular access links). In the cloud, attackers can also find ways to exhaust resources that should be used for guest systems.

2.4.3 Impact of Security on Performance

There is always a tradeoff between security and performance in computer systems and services. In mobile cloud computing, security measures typically require extra protection procedures to encrypt offloaded mobile applications and data, to authenticate mobile devices and users, and to maintain trustable relations between mobile devices and cloud through exchanging and updating credentials. Hence, security requests more resources for keeping credentials and performing security functions, and thus adds additional overhead in terms of processing time and power consumption.

3 Threats Against Mobile Cloud Computing

Mobile cloud computing architectures are vulnerable to a different and wider range of attacks than traditional computing systems [4]. This is primarily due to the following reasons:

1. *Limited resources*: Despite major technological advances, mobile devices are still constrained by computational, networking, storage and power capabilities in comparisons to regular computers. This prevents these devices from running the necessary security applications and leaves them vulnerable to attacks. For example, monitoring tools such as packet sniffing tools and intrusion detection systems may consume extensive resources that would overwhelm the whole system and shorten its lifetime. Thus an inherent tradeoff between security and performance is present.
2. *Multiple locations for storage and computation*: Data from mobile devices are often uploaded and stored in the cloud. Also, extensive computation is offloaded and executed in the cloud. This makes the data and code vulnerable to attacks at multiple locations (e.g., the mobile device, the cloudlet and the cloud). Moreover, the links utilized in transferring the data and code could be targeted by attackers. In essence the attacker can target the weakest resource in the chain to access confidential information or compromise the integrity of the computation. The involvement of third-parties complicates these issues further.
3. *Mobility as a Service*: In mobile cloud computing, users have the capability to move their computation between different public clouds in addition to some cloud providers that offer mobility as a service for its users [1]. Malicious users can infer cloud specifics (e.g., cloud topology, the bottleneck links and the load on the machines) and launch sophisticated attacks that target specific resources (e.g., bottleneck links) [41]. Thus mobility in cloud architectures brings new sets of threats to the table.

In this section, we expose threats against mobile cloud computing architectures focusing on attacks that target confidentiality, integrity and availability.

3.1 Mobile Devices

When it comes to privacy, mobile devices carry valuable personal information (e.g., photos, contact lists and communication history) as well as critical application data (e.g., health records and credit card information). When a mobile device gets in the wrong hands (even for a short duration), attackers can perform forensics analysis and extract such information. There are many legitimate tools for forensics examiners that are publicly available and can be used for malicious intent [39, 40]. Another example is the Joint Test Access Group (JTAG) attacks [54] in which attackers can access testing and debugging functionalities on the device to extract confidential information.

Perhaps one of the major incidents concerning user's privacy was that with Carrier IQ in which it was reported that Carrier IQ software captured every keystroke, location and other data on a wide range of smart phones. Collected information was transmitted and made available to Carrier IQs customers [21].

In recent years, malware has proliferated that threatens the privacy of the users through exploiting the operating system. In [48], it was demonstrated that through a drive-by download, an attacker can potentially steal the short message service (SMS) database from an iPhone 3GS. On the Android platform, GingerMaster is malware that is packaged into legitimate applications. The malware runs in the background collecting information and sending it to a remote server [33]. RootMaster is another type of malware that would gain root privilege on an Android device, connect to a command and control server and even install additional malware [34].

In general, if an attacker gains root permissions on a mobile device through malware, one could impact the computational integrity of the operating systems as well as of other applications (e.g., provide incorrect sensing information or results). To the best of our knowledge, we are not aware of specific threats that achieved this. However, it suffices to mention that the above malware that can control a mobile device can be orchestrated to compromise other applications and impact the integrity of the computation performed.

When it comes to availability, wireless communication on mobile devices is inherently vulnerable to interference and jamming attacks [47, 68]. Attackers can acquire hardware jammers to prevent the device from receiving the intended signals. Since many mobile devices rely on a wireless interface for communication, such interface may be subjected to a wide range of attacks on the Media Access Control (MAC) layer [37, 55, 63, 67] as well as traditional DoS attacks in which attackers send a large amount of traffic that would saturate the wireless link, preventing the transmission of other information on that interface. Moreover, if it is known that a mobile device is implementing a particular packet sniffing application or an intrusion detection system, an attacker may subject the device to a malicious traffic stream that triggers specific rules to be applied which, in turn, overwhelms the resources (e.g., rules that try to match strings). Another form of attack that targets the availability is through malware that executes junk instructions just to drain the power of the mobile device.

3.2 Cloudlets

When a mobile device uploads data/code on the cloud/cloudlet [64] for storage/processing, it becomes vulnerable to privacy issues and computational integrity attacks. For example, many architectures rely on creating an image clone of the mobile device on the cloud [12]. Execution can then occur on the device or in the virtualized cloud environment. According to recent report [52], mobile and cloud attacks will

be dramatically increased because users and attackers shift to mobile and cloud in 2013. Georgia Tech researchers [38] also predicted that cloud botnets and mobile attacks will be the biggest cloud issues in 2013.

The integrity of the computation is directly affected by the technologies adopted in the cloud. Virtualization has been a key player in the development of cloud computing architectures. Users upload their images as virtual machines to be executed in the cloud. To save resources, multiple virtual machines may share the same server. This causes a security mismatch in the sense that users would like to keep their data and code private while the cloud needs to ensure the security and safety of the code being executed on their physical infrastructure [11]. Dynamic Data Kernel Rootkit attacks may target the VMM and control the execution of the host operating system, thus affecting all other virtual machines running on top [20, 61]. Moreover, it has been shown that certain types of malware can, in fact, detect that they are being executed in a virtual environment and would change their behavior accordingly [50].

Beyond the traditional availability attacks that can occur on access links to the cloud, new forms of threats started to surface. In [41], the authors describe an attack scheme in which attackers can carry out DoS attacks through targeting specific links within the cloud architecture and saturating them. The attack scheme starts by inferring the network topology within the cloud and then choosing specific hosts that share common bottleneck links and sends large amounts of traffic between them. The authors demonstrated this attack scheme on one cloud infrastructure. Moreover, the ability of the users to move computation between clouds/cloudlets seems to open backdoors for attacks to be mounted.

3.3 Ad Hoc Mobile Clouds

In this section, we study threats against code distribution mechanisms that enable multiple parties to execute code segments for specific users. These parties could be other mobile users (ad hoc), cloudlet components, or a hybrid.

Many applications require the help of nearby devices in executing code segments. Such applications arise in scenarios in which executing the whole application on a single device is not feasible due to various constraints such as: (1) limited battery power constraints, since a single device may not have enough battery life to execute the whole application to completion; (2) limited infrastructure access, since a single device may not have access to the infrastructure (e.g., dead zones, remote areas) or there is a high cost incurred (e.g., exceeding the cellular data allowance); (3) different working data sets [56], since nearby mobile devices may operate on local data stored on them (e.g., photos captured within a specific time frame) or acquire data from the environment around them (e.g., collecting sensor information).

Due to the reliance on a distributed manner in executing code segments, the existence of malicious nodes should be considered. Malicious nodes can also collude to reverse-engineer code segments in order to reveal the functionality and

capability of the code. If the malicious nodes can share the code and the data they are operating on, the privacy of the requesting user is violated. Although in [42], the security threats of colluding applications have been studied, to the best of our knowledge, we have not yet seen colluding threats that target code partitioning. As for the integrity of the executing segments, when users ask nearby devices to execute code segments on their behalf, they run the risk of users not responding or responding with incorrect results. The requesting device cannot easily verify the correctness of the results, unless by adding a layer of redundancy (multiple devices are asked to execute the same code segments and the results are compared). Unfortunately, this adds a lot of overhead and the process of managing the segments may be itself subjected to a new set of clever attacks. Jamming attacks or Denial of service (DoS) attacks on the links between the mobile nodes can impact the code distribution modules. In particular, a subset of the mobile devices may not receive their assigned code segments. This prevents the requesting users from receiving the results and may trigger additional work to redistribute the segments that have not been executed, leading to longer latencies. Another threat that targets the availability of the systems is for malicious users to distribute code segments that causes mobile devices to drain their power (e.g., code that requires a lot of CPU computation and communication).

4 Mobile Cloud Computing Security Measures

In this section, we overview recently proposed security solutions for mobile computing in the cloud. The summary will be focused on the security aspects closely related to computing. Security works, such as vulnerability assessment [36], communication security [4], interface and session security [59], verifiable accounting [58], and so on, are not included in this section.

4.1 Mobile Devices

As discussed in [4], mobile devices are threatened by various attacks targeting their hardware, operating systems, applications software, communication links, back-end systems, and users. Although all these threats concern the security of mobile devices, this section will discuss computing-related security measures that ensure confidentiality and integrity of mobile computing in the cloud. In particular, the security of the device storage and operating system have direct impacts on mobile computing in the cloud and are the trust base of newly proposed security measures.

4.1.1 Mobile Device Storage Security

The storage of a mobile device is usually made of an on-board memory and a plugged-in memory. The on-board memory stores the code and data of the running software. The plugged-in memory includes non-volatile flash storage and subscriber identity module (SIM) card storage. The stored security-critical data includes, but is not limited to, personal information in the flash, device credential in the SIM card, and runtime keys in the on-board memory. In cloud computing, security-critical code and data could be mirrored in the cloud as well. Because mobile devices can be lost or stolen, they become a weak point in the complete system of mobile cloud computing. Attackers can obtain access to the internals of stolen devices without the need to hack into the cloud. Hence, securing the device storage is a must for protecting data confidentiality and personal privacy.

Attackers can obtain access to the code and data stored in on-board memory via JTAG functionalities [32] or through directly tapping into the circuit board [16]. The security measures in protecting on-board memory include disabling JTAG functions and implementing secure storage and secure access to user I/O facilities following the recommendation of Open Mobile Terminal Platform (OMTP) Advanced Trusted Environment [44].

Attackers can obtain access to the code and data stored in plugged-in memory via forensic analysis [65]. The protection measures rely on the encryption of either the data or the memory. The security of encryption keys needs Trusted Platform Module (TPM), which is either a stand-alone chip in the device or a part of the functionality of the SIM card.

4.1.2 Operating System Security

One major advancement in mobile phone technology is the phone's mobile operating system (MOS) that fully utilizes the advancement of the phone's hardware and enables a large variety of mobile applications. Currently, the MOS functionalities are quite close to regular computer operating system (OS). Securing MOS is the fundamental measure to ensure the confidentiality and the integrity of running mobile applications in the cloud.

The security of MOSs can be achieved with kernel hardening, security control over applications, and secure updating and installation. Because MOSs are derived from regular OSs, their kernels are hardened with similar security techniques as regular OSs, including address space layout randomization and non-executable data memory. Mandatory Access Control (MAC) can also be implemented in MOSs to enhance the overall security.

Learned from hard lessons in regular computer, MOSs should be designed with security control over applications based on the principles of lease privileges and separation of duty. For example, each application should run with only the needed privileges for completing a job. Although many applications request more privileges than needed during installation in the Android, users can install additional tool-kits

to disable unwanted privileges for each application. In Android, each application runs in a process isolated from other processes through virtualization of the phone so that a compromising process will not endanger other processes or the operating system.

Most modern MOSs support updating the system remotely to update or patch drivers, the OS modules and the installed applications. Remote updating in MOSs can be protected in the same fashion as with regular OSs [5]. Installing applications, however, is a more complicated process. In Android, a list of privileges come with each application to be installed. Accepting the list is the only way to install the application. However, the privileges in the list usually give the application access to personal data without any additional mechanisms to actually inspect how the application uses the data. A category of security measures have been proposed to disable unwanted privileges [17] or assign only needed permissions at run time [46]. Another category of security measures is to profile and track the information flow of applications and discover the critical points in time when applications access sensitive information or are about to expose such information [18, 28].

4.2 Cloud Servers

As discussed in Section 3, the cloud can be threatened by many different attacking techniques that target different components of the cloud. Various security solutions have been proposed in the past to address these threats in the cloud. In this section, we discuss the security measures that are closely related to the security of cloud computing and that are applicable to mobile computing in the cloud as well. Because the traditional cloud is built upon the client-server model, the security measures in this section focus on the cloud side. In particular, we discuss the defense mechanisms to ensure confidentiality, integrity and availability of computing in the cloud servers.

As illustrated in Fig. 1, the system components in a cloud server include the guest systems, the VMM and the host system. Accordingly, we classify the security measures into three categories. The first category is on the security of computation in a guest system. The second category focuses on how a VMM manages the guest systems. The third category is on the security of user-controlled operations that may affect the security of the cloud.

4.2.1 Guest Systems

As more and more services are transferred to the cloud, each user's personal information is transferred to the cloud as well. When users execute their applications in guest systems with their personal information, the privacy and integrity of the computation inside the cloud are concerns that must be addressed to convince the users to use the cloud services.

A straightforward approach is access control that enforces security policies so that only authorized entities and processes can access data and applications. However, if the data and the applications are stored in the clear, administrators of the cloud can access them or even change the security policies [45]. Therefore, cryptographic solutions are desired so that data and applications are encrypted in the cloud. One promising cryptographic solution is the fully homomorphic encryption [23,24,43], where data is not only encrypted in storage but also in computation. With fully homomorphic encryption, computation is performed directly over encrypted data, and the encrypted results are returned to the users. Then, the users decrypt the results to obtain the actual results as if the computation was performed over clear data. Hence, the encryption ensures that no one in the cloud could peak at the users' personal data. The hurdle of fully homomorphic encryption solutions in practice is the computational inefficiency in that it is much slower than classical encryptions and regular data operations. Another issue is that the encryption only ensures confidentiality and privacy but not the integrity of computation. It is hard for the cloud to prove that the result was not manipulated during the computation based on a fully homomorphic encryption.

In another work [15], the authors propose to use some well-established cryptographic solutions to achieve the desired security of computation but on the client side. Personal data is encrypted as cryptographic commitments, which not only hide the data but also can be used later to verify the data in the cloud. Nevertheless, the computation is not performed in the cloud but rather on the client. When the cloud needs to process user's data, it sends the commitments to the user and requests the user to perform some computation. Then, the user sends the results together with non-interactive zero-knowledge proofs back. The cloud verifies the results with the commitments and the zero-knowledge proofs to validate the integrity of the computation. This solution is an interim solution to the needs of real-world scenarios before the fully homomorphic encryption can well satisfy the security needs.

4.2.2 Virtual Machine Managers (VMM)

The VMM is the component that manages multiple guest systems in a cloud server. The security of VMM usually refers to two aspects. One is the security among multiple guest systems, i.e., a compromised guest system shall not endanger other guest systems. The other is the security of the guest systems themselves, i.e., a guest system shall perform its own computation as expected.

To achieve the first security goal, isolation is the major mechanism, by which VMM places boundaries between computational resources of different guest systems. The boundary could be physical, i.e., each guest system is hosted on different machines that are physically locked in different locations. In practice, isolation is virtualized so that cloud providers can flexibly allocate fine-grained and requested computational resources to the guest systems. The problem of virtualized isolation is that a guest system may interfere with another guest system. Although isolation is in place, multiple guest systems in the same computer are still sharing computational

resources, such as CPU cores, caches, memory, and I/O channels. Among the resources, cache is the most complicated factor to isolate, because it is inside the CPU core and often not directly controlled by QoS-provision VMM. A cache-aware isolation solution was proposed in [51] to address the sharing issue of the last level cache (LLC). The solution places the cores that share the same LLC into one core group and then assigns each core group to at most one guest system. Thereby, a guest system will not compete for caches with other guest systems. An obvious drawback of this solution is that the granularity of the computational resource is at the level of the core group. If a resource of a core group is under utilized with a guest system, this resource cannot be assigned to other guest systems.

The second security goal requires integrity checks on the running guest systems. Because applications in the guest systems may access the Internet and thus experience attacks as do regular computer systems, it is important to ensure that the guest systems are not compromised during execution. Since the cloud is built using virtualization technologies, it is fairly convenient to implement the integrity check as a part of the functionality of the VMM. Various integrity check solutions have been proposed. One type of integrity check solution is virtual machine introspection [11, 22], which is an intrusion detection module in the VMM that monitors the state of the processors, memory and disks in the guest systems to detect intrusive actions. A second type of integrity check solution is to check the execution of the guest system's code. Before executing the code, it is stored in a protected memory and fetched from that memory [53], or the control flow of the code is verified first [49]. These solutions need the behavior of the guest systems, which must be stored in the VMM so that it can compare the running guest system with the expected behavior.

4.2.3 User-Controlled Security Measures

Mobile users are involved in cloud computing in two kinds of operations: VM image dissemination, and end-user configuration. Because users create, upload and retrieve VM images, their dissemination operations may disclose their own personal information. They may also retrieve malicious images from the cloud. Note that the role of the cloud provider is just to provide images to clients with the authorization of the owners of the images. Hence, securely managing images is an important task in the cloud. Mirage [66] is a VM image management system, and consists of four components. The first component is an access control framework that regulates the sharing of VM images under the discretion of the image's owner. The second component consists of image filters that are applied to remove or hide sensitive information when sharing images. The third component is a provenance tracking mechanism that tracks the derivation history of an image when a user revises the image and generates a new one. The last component is a set of repository maintenance services that detect and fix vulnerabilities in images.

In the cloud, users are allowed to group machines and enact rules to control communication among groups. In this manner, users can set up groups with different security levels and provide customized services in the cloud. However,

such end-user configuration may put users' information at risk, considering the complexity of the cloud infrastructure and services. Therefore, it is necessary to assess the security of the end-user configuration to protect users' information. In [6], a configuration audit approach was proposed for multi-tier cloud infrastructure. The approach models the security policy rules as an information flow graph, in which each vertex represents a repository of information and each edge represents the information flow. Using the graph, the approach conducts reachability analysis according to the security policy rules. A violation is detected when an information flow path exists in the graph but is not specified by the policy rules.

4.3 Mobile Clouds

Security measures in traditional cloud settings can be applied to address a significant portion of the security needs for mobile cloud computing, because the design and management of cloud servers are usually not affected by the type of the clients. Nevertheless, as discussed in Section 2, new architectures and computing models have been proposed for mobile computing in the cloud, which raise new and unique security needs of mobile computing. In particular, new security measures were developed in the following aspects of mobile computing: (1) secure code partitioning and offloading, (2) mobile user authentication, and (3) secure mobile data management. The security of these three aspects not only provide the integrity and confidentiality to mobile computing in the cloud, but also take advantage of the features present in mobile devices.

4.3.1 Secure Code Partitioning and Offloading

Code partitioning and offloading were common functionalities in all of the three mobile cloud computing architectures outlined above. Hence, securing them is an indispensable aspect of securing elastic applications in mobile cloud computing. A secure elastic framework [27, 69] was proposed to address this issue. The framework consists of four components: secure installation, module authentication, secure migration, and permission authorization. Although the framework is designed to weblet-based elastic applications, the principal ideas of its security components can be applied to other modular elastic applications.

- The *secure installation component* ensures that genuine applications are installed on the mobile devices and that applications have signed hashes. During installation, the hashes of applications are checked to ensure the integrity of applications. After successful verification, the applications are registered with elastic services.
- The *module authentication component* enables one module of an application to authenticate another module that belongs to the same application. Because modules of the same application may be migrated and executed at different

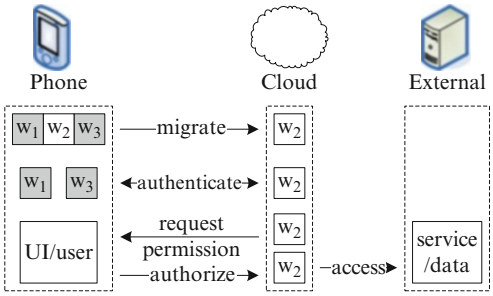

Fig. 5 Secure execution of an elastic application

locations, dependent modules need to be authenticated in the sequence of execution to ensure that the genuine modules of the same applications are executed regardless of the locations of execution. Upon executing the modules, the manager of elastic applications generates a pair of session key and session secret associated with each module. The pair of session key and secret are then used for authentication and secure communication among modules.

- The *secure migration component* ensures the security of the migration process. When a module needs to be migrated, the module saves its state and enters the migration state. During migration, the pair of key and secret associated with the module are delivered with the module to the destination for authentication. After verification, the migrated module resumes its execution from the saved state.
- The *permission authorization component* assigns different permissions to the modules of an application. Because the functionality and the location of modules are different within the same application, they should be authorized with different permissions based on their tasks. One proposed approach is that modules in the cloud, if requesting access to sensitive data, should forward such requests to the mobile device to obtain credentials from the device, and then use the credentials to authenticate and obtain authorization. Another approach is that the modules in the cloud initiate an authentication request to access sensitive data, but ask the mobile device to complete the authentication process and forward the resulting authenticated session to the modules in the cloud.

An example of executing an elastic application with the four components is illustrated in Fig. 5, where the application has three weblets w_1, w_2 and w_3. Upon execution, w_2 is migrated to the cloud for execution by the migration component. During execution, w_2 authenticates with w_1 and w_3 using the authentication component. At the time when w_2 needs to access an external server, w_2 requests authorization through the authorization component. The user may be prompted to authorize the access through the phone's user interface (UI). If granted, w_2 proceeds to access the external server.

The secure elastic framework emphasizes on the management, migration, and the execution of the offloaded modules of an application. The main security goal is the integrity of the elastic application when its modules are executed at different locations. However, the framework is weak when it comes to ensuring the security of

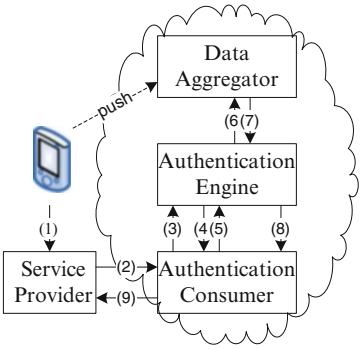

Fig. 6 Mobile client authentication

the code partitions themselves. The current focus of code partition is on optimizing the overhead of migration and execution with partitioned code modules. Security is not considered in the model of code partitioning so far. In [8], the confidentiality and integrity of the partitioned programs are studied by using security labels annotated in programs. Privacy of sensitive data in cloud computing is also studied in [35,70]. However, research on similar security issues for elastic mobile applications in the cloud is lacking.

4.3.2 Mobile User Authentication

Mobile cloud computing requires secure authentication methods to prevent data theft. Typical mobile authentication approaches rely on identity information of ownership factors (e.g., mobile devices and security tokens), knowledge factors (e.g., password, pass phrase, or personal identification number (PIN), and challenge questions and answers), and inherence factors (e.g., fingerprint and other biometric identifiers). Some unique and critical authentication information could be stored on mobile devices. However, the identity information is insecure because mobile devices could be stolen, and/or attackers could steal identity information. To solve this problem, Song et al. [9] proposed a new authentication principle in the TrustCube infrastructure that is an end-to-end infrastructure to control the access of authenticated clients by using the history of authenticated actions and behaviors of mobile clients (users). Their approach is built on the TrustCube platform [60], which provides client authentication services in clouds with integration of various authentication methods. The authentication framework for mobile clients [9], as illustrated in Fig. 1 in [9], consists of three components in the cloud: *Data Aggregator*, *Authentication Engine*, and *Authentication Consumer*. The *Data Aggregator* collects past actions of the mobile devices. The *Authentication Engine* obtains data from the *Data Aggregator* and makes authentication decisions. The *Authentication Consumer* provides policies to the *Authentication Engine*.

We illustrate the mobile authentication procedure in Fig. 6. The client device regularly pushes the mobility traces to the *Data Aggregator* to reduce risk of

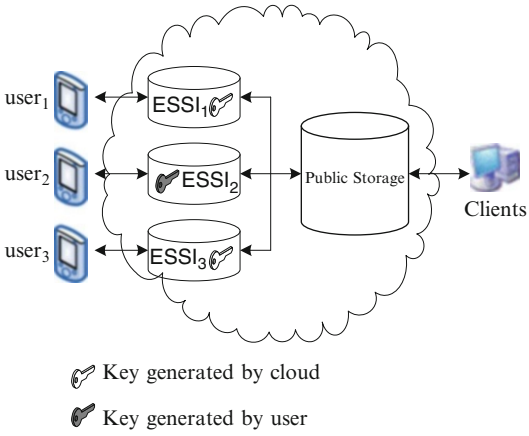

Fig. 7 Secure data processing and management in mobile cloud computing

identity theft. When the client intends to access a cloud service (*step (1)*), the request of authentication will be forwarded to the *Authentication Consumer* (*step (2)*), which redirects the request to the *Authentication Engine* (*step (3)*). The *Authentication Engine* obtains the policy for this access from the *Authentication Consumer* (*steps (4) and (5)*), and queries the *Data Aggregator* about the client device (*step (6)*). The *Data Aggregator* furnishes past authenticated actions and behaviors of a mobile client to the *Authentication Engine* (*step (7)*). The *Authentication Engine* exchanges a secret with the *Authentication Consumer* during authentication in order to later verify the authentication result. The *Authentication Engine* determines the authentication result and furnishes it to the *Authentication Consumer* (*step (8)*). Finally, the *Authentication Consumer* notifies the service provider concerning the access permission result such as *access acceptance* or *access denial* (*step (9)*).

4.3.3 Secure Mobile Cloud Computing

In mobile cloud computing, the information collected by mobile devices is processed, stored, and provided to other users as a part of cloud services. The images of mobile devices could be stored in the cloud to augment the functionalities of the devices as well. Confidentiality and privacy of this information is a concern which is gaining much attention. A secure data processing framework for mobile cloud computing [30] is proposed to address this concern.

We illustrate a secure mobile data processing and management in Fig. 7. The cloud is divided into a public service domain and a trusted domain. The public domain stores information generated from the mobile devices that can be provided to the public. The trusted domain is made of extended semi-shadow images (ESSIs) [29, 30], which are clones of the mobile devices in a secure storage. To ensure the security of a user's data, the cloud generates keys for each user to encrypt data in ESSIs. When the public service needs the data of a user, the data will

be obtained from the ESSI of the user. The data will be preprocessed to remove identity information before being used in the public domain. However, security concerns on the encrypted data in ESSIs still remain since the keys are generated and controlled by the cloud providers. Therefore, the secure data processing framework [30] applies a multi-tenant data management scheme that partitions data into two security levels: critical data and normal data. The normal data is stored and secured by the data encryption keys as aforementioned. The critical data is encrypted with keys generated by the owners. Hence, access to critical data needs the authorization of the owners. The data processing in ESSI is built on the security capability model. Three capabilities are defined: cloud root, user root, and auditing root. The user root encrypts and decrypts data in ESSI.

5 Conclusion

In this chapter, we reviewed the latest research and development of secure mobile computing in the cloud. We first discussed three representative cloud architectures designed to support new mobile computing models in the cloud. We showed that new features can be realized to extend the computing capabilities of mobile devices when the advantages of the mobile devices and cloud computing are integrated into one system. We then studied a wide range of threats against the availability, privacy and integrity of mobile cloud computing architectures. We showed that attackers can target and exploit a much wider range of resources/protocols in a mobile cloud computing environment when compared to traditional client-server architectures. Finally, we summarized newly proposed defense mechanisms that ensure the security of mobile cloud computing architectures and their applications. We showed that securing mobile computing in the cloud needs not only the applicable defenses in traditional cloud, but also the security solutions designed specially for mobile cloud architectures.

Acknowledgements This material is based upon work partially supported by the One-Time Research Support Program at Texas State University-San Marcos, the National Science Foundation (NSF) grant CNS-1149397, the Air Force Office of Scientific Research (AFOSR)/the Air Force Research Laboratory (AFRL) Visiting Faculty Research Program (VFRP) extension grant LRIR 11RI01COR.

References

1. Baliga, A., Chen, X., Coskun, B., de los Reyes, G., Lee, S., Mathur, S., Van der Merwe, J.E.: VPMN: virtual private mobile network towards mobility-as-a-service. In: Proceedings of the 2nd International Workshop on Mobile Cloud Computing and Services, MCS'11, Washington, DC, pp. 7–12. ACM, New York (2011). doi:10.1145/1999732.1999735
2. Barrera, D., Kayacik, H.G., van Oorschot, P.C., Somayaji, A.: A methodology for empirical analysis of permission-based security models and its application to Android. In: Proceedings

of the 17th ACM Conference on Computer and Communications Security, CCS'10, Chicago, pp. 73–84. ACM, New York (2010). doi:10.1145/1866307.1866317. http://doi.acm.org/10.1145/1866307.1866317
3. Barrera, D., Clark, J., McCarney, D., van Oorschot, P.C.: Understanding and improving app installation security mechanisms through empirical analysis of Android. In: Proceedings of the 2nd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM'12, Raleigh, pp. 81–92. ACM, New York (2012). doi:10.1145/2381934.2381949
4. Becher, M., Freiling, F.C., Hoffmann, J., Holz, T., Uellenbeck, S., Wolf, C.: Mobile security catching up? Revealing the nuts and bolts of the security of mobile devices. In: Proceedings of the 2011 IEEE Symposium on Security and Privacy, SP'11, Oakland, pp. 96–111. IEEE Computer Society, Washington, DC (2011). doi:10.1109/SP.2011.29
5. Bellissimo, A., Burgess, J., Fu, K.: Secure software updates: disappointments and new challenges. In: Proceedings of the 1st USENIX Workshop on Hot Topics in Security, HOTSEC'06, Vancouver, pp. 37–43. USENIX Association, Berkeley (2006)
6. Bleikertz, S., Schunter, M., Probst, C.W., Pendarakis, D., Eriksson, K.: Security audits of multi-tier virtual infrastructures in public infrastructure clouds. In: Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop, CCSW'10, Chicago, pp. 93–102. ACM, New York (2010). doi:10.1145/1866835.1866853
7. Chaudhuri, A.: Language-based security on Android. In: Proceedings of the ACM SIGPLAN 4th Workshop on Programming Languages and Analysis for Security, PLAS'09, Dublin, pp. 1–7. ACM, New York (2009). doi:10.1145/1554339.1554341
8. Chong, S., Liu, J., Myers, A.C., Qi, X., Vikram, K., Zheng, L., Zheng, X.: Secure web applications via automatic partitioning. SIGOPS Oper. Syst. Rev. **41**(6), 31–44 (2007). doi:10.1145/1323293.1294265
9. Chow, R., Jakobsson, M., Masuoka, R., Molina, J., Niu, Y., Shi, E., Song, Z.: Authentication in the clouds: a framework and its application to mobile users. In: Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop, CCSW'10, Chicago, pp. 1–6. ACM, New York (2010). doi:10.1145/1866835.1866837
10. Christensen, J.H.: Using RESTful web-services and cloud computing to create next generation mobile applications. In: Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications, OOPSLA'09, Orlando, pp. 627–634. ACM, New York (2009). doi:10.1145/1639950.1639958
11. Christodorescu, M., Sailer, R., Schales, D.L., Sgandurra, D., Zamboni, D.: Cloud security is not (just) virtualization security: a short paper. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, CCSW'09, Chicago, pp. 97–102. ACM, New York (2009). doi:10.1145/1655008.1655022
12. Chun, B.G., Maniatis, P.: Augmented smartphone applications through clone cloud execution. In: Proceedings of the 12th Conference on Hot Topics in Operating Systems, HotOS'09, Monte Verita, pp. 1–5. USENIX Association, Berkeley (2009)
13. Chun, B.G., Maniatis, P.: Dynamically partitioning applications between weak devices and clouds. In: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, MCS'10, San Francisco, pp. 7:1–7:5. ACM, New York (2010). doi:10.1145/1810931.1810938
14. Cuervo, E., Balasubramanian, A., Cho, D.k., Wolman, A., Saroiu, S., Chandra, R., Bahl, P.: MAUI: making smartphones last longer with code offload. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, MobiSys'10, San Francisco, pp. 49–62. ACM, New York (2010). doi:10.1145/1814433.1814441
15. Danezis, G., Livshits, B.: Towards ensuring client-side computational integrity. In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, CCSW'11, Chicago, pp. 125–130. ACM, New York (2011). doi:10.1145/2046660.2046683
16. Drimer, S., Murdoch, S.J., Anderson, R.: Thinking inside the box: system-level failures of tamper proofing. In: Proceedings of the 2008 IEEE Symposium on Security and Privacy, SP'08, Oakland, pp. 281–295. IEEE Computer Society, Washington, DC (2008). doi:10.1109/SP.2008.16

17. Enck, W., Ongtang, M., McDaniel, P.: On lightweight mobile phone application certification. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS'09, Chicago, pp. 235–245. ACM, New York (2009). doi:10.1145/1653662.1653691
18. Enck, W., Gilbert, P., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N.: TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, OSDI'10, Vancouver, pp. 1–6. USENIX Association, Berkeley (2010)
19. Fahl, S., Harbach, M., Muders, T., Baumgärtner, L., Freisleben, B., Smith, M.: Why eve and mallory love Android: an analysis of Android SSL (in)security. In: Proceedings of the 19th ACM Conference on Computer and Communications Security, CCS'12, Raleigh, pp. 50–61. ACM, New York (2012). doi:10.1145/2382196.2382205
20. Florio, E.: symantec.com, when malware meets rootkits. http://goo.gl/WdznF
21. forbes.com: Phone rootkit maker carrier IQ may have violated wiretap law in millions of cases. http://goo.gl/P3NJg
22. Garfinkel, T., Rosenblum, M.: A virtual machine introspection based architecture for intrusion detection. In: Proceedings of the 10th Network and Distributed Systems Security Symposium, NDSS'03, San Diego, pp. 191–206 (2003)
23. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC'09, Bethesda, pp. 169–178. ACM, New York (2009). doi:10.1145/1536414.1536440
24. Gentry, C.: Computing arbitrary functions of encrypted data. Commun. ACM **53**(3), 97–105 (2010). doi:10.1145/1666420.1666444
25. Gilbert, P., Chun, B.G., Cox, L.P., Jung, J.: Vision: automated security validation of mobile apps at app markets. In: Proceedings of the 2nd International Workshop on Mobile Cloud Computing and Services, MCS'11, Bethesda, pp. 21–26. ACM, New York (2011). doi:10.1145/1999732.1999740
26. Giurgiu, I., Riva, O., Juric, D., Krivulev, I., Alonso, G.: Calling the cloud: enabling mobile phones as interfaces to cloud applications. In: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware, *Middleware'09*, Urbana, vol. 5, pp. 5:1–5:20. Springer, New York (2009)
27. He, S., Guo, L., Guo, Y.: Elastic application container. In: Proceedings of the 12th IEEE/ACM International Conference on Grid Computing, GRID'11, Lyon, pp. 216–217. IEEE Computer Society, Washington, DC (2011). doi:10.1109/Grid.2011.35
28. Hornyack, P., Han, S., Jung, J., Schechter, S., Wetherall, D.: These aren't the droids you're looking for: retrofitting Android to protect data from imperious applications. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS'11, Chicago, pp. 639–652. ACM, New York (2011). doi:10.1145/2046707.2046780
29. Huang, D., Zhang, X., Kang, M., Luo, J.: MobiCloud: building secure cloud framework for mobile computing and communication. In: Proceedings of 5th IEEE International Symposium on Service Oriented System Engineering, SOSE'10, Nanjing, pp. 27–34. IEEE Computer Society, Washington, DC (2010). doi:10.1109/SOSE.2010.20
30. Huang, D., Zhou, Z., Xu, L., Xing, T., Zhong, Y.: Secure data processing framework for mobile cloud computing. In: Proceedings of the IEEE Conference on Computer Communications Workshop, Shanghai, pp. 614–618 (2011). doi:10.1109/INFCOMW.2011.5928886
31. Huerta-Canepa, G., Lee, D.: A virtual cloud computing provider for mobile devices. In: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, MCS'10, San Francisco, pp. 6:1–6:5. ACM, New York (2010). doi:10.1145/1810931.1810937
32. Jack, B.: blackhat.com, exploiting embedded systems. http://goo.gl/oz7Vs (2006)
33. Jiang, X.: ncsu.edu, GingerMaster: first Android malware utilizing a root exploit on Android 2.3 (Gingerbread). http://goo.gl/uvTFT
34. Jiang, X.: ncsu.edu, security alert: new RootSmart Android malware utilizes the GingerBreak root exploit. http://goo.gl/ZTxpg

35. Ko, S.Y., Jeon, K., Morales, R.: The HybrEx model for confidentiality and privacy in cloud computing. In: Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'11, Portland, pp. 1–5. USENIX Association, Berkeley (2011)
36. Kupsch, J.A., Miller, B.P., Heymann, E., César, E.: First principles vulnerability assessment. In: Proceedings of the 2010 ACM Workshop on Cloud Computing Security, CCSW'10, Chicago, pp. 87–92. ACM, New York (2010). doi:10.1145/1866835.1866852
37. Law, Y.W., Palaniswami, M., Hoesel, L.V., Doumen, J., Hartel, P., Havinga, P.: Energy-efficient link-layer jamming attacks against wireless sensor network MAC protocols. Trans. Sens. Netw. **5**(1), 6:1–6:38 (2009). doi:10.1145/1464420.1464426
38. Lee, W., Rotoloni, B.: Emerging cyber threats report 2013. Technical report, Georgia Institute of Technology (2012)
39. Lessard, J., Kessler, G.: Android forensics: simplifying cell phone examinations. Small Scale Digit. Device Forensics J. **4**(1), 1–12 (2010)
40. linuxsleuthing.blogspot.com, Linux Sleuthing: iPhone forensics tools. http://goo.gl/Wc31M
41. Liu, H.: A new form of DoS attack in a cloud and its avoidance mechanism. In: Proceedings of the 2010 ACM Workshop on Cloud Computing Security, CCSW'10, Chicago, pp. 65–76. ACM, New York (2010). doi:10.1145/1866835.1866849
42. Marforio, C., Francillon, A., Capkun, S.: osti.gov, application collusion attack on the permission-based security model and its implications for modern smartphone systems. http://goo.gl/0Csm2
43. Micciancio, D.: A first glimpse of cryptography's holy grail. Commun. ACM **53**(3), 96–96 (2010). doi:10.1145/1666420.1666445
44. omtp.org: OMTP advanced trusted environment. http://goo.gl/Nzf6p (2009)
45. Ongtang, M., Butler, K., McDaniel, P.: Porscha: policy oriented secure content handling in Android. In: Proceedings of the 26th Annual Computer Security Applications Conference, ACSAC'10, Austin, pp. 221–230. ACM, New York (2010). doi:10.1145/1920261.1920295
46. Ongtang, M., McLaughlin, S., Enck, W., McDaniel, P.: Semantically rich application-centric security in Android. Secur. Commun. Netw. **5**(6), 658–673 (2012). doi:10.1002/sec.360
47. Pelechrinis, K., Iliofotou, M., Krishnamurthy, V.: Denial of service attacks in wireless networks: the case of jammers. IEEE Commun. Surv. Tutor. **13**(2) (2011). doi:10.1109/SURV.2011.041110.00022
48. Portnoy, A.: tippingpoint.com, Pwn2POwn 2010. http://goo.gl/XLJN
49. Quynh, N.A., Takefuji, Y.: Towards a tamper-resistant Kernel rootkit detector. In: Proceedings of the 2007 ACM Symposium on Applied Computing, SAC'07, Seoul, pp. 276–283. ACM, New York (2007). doi:10.1145/1244002.1244070
50. Raffetseder, T., Kruegel, C., Kirda, E.: Detecting system emulators. In: Proceedings of the Information Security, Valparaíso, pp. 1–18 (2007)
51. Raj, H., Nathuji, R., Singh, A., England, P.: Resource management for isolation enhanced cloud services. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, CCSW'09, Chicago, pp. 77–84. ACM, New York (2009). doi:10.1145/1655008.1655019
52. Ramsey, R.: tmcnet.com, as users shift to mobile and cloud, so will attackers: cybercrime in 2013. http://goo.gl/MLeuk (2012)
53. Riley, R., Jiang, X., Xu, D.: Guest-transparent prevention of Kernel rootkits with VMM-based memory shadowing. In: Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection, RAID'08, Cambridge, pp. 1–20. Springer, Berlin/Heidelberg (2008). doi:10.1007/978-3-540-87403-4_1
54. Rosenfeld, K., Karri, R.: Attacks and defenses for JTAG. IEEE Des. Test **27**(1), 36–47 (2010). doi:10.1109/MDT.2010.9
55. Sang, L., Arora, A.: Capabilities of low-power wireless jammers. In: Proceedings of INFOCOM, Rio de Janeiro (2009). doi:10.1109/INFCOM.2009.5062185
56. Satyanarayanan, M.: Mobile computing: the next decade. SIGMOBILE Mobile Comput. Commun. Rev. **15**(2), 2–10 (2011). doi:10.1145/2016598.2016600
57. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. **8**(4), 14–23 (2009). doi:10.1109/MPRV.2009.82

58. Sekar, V., Maniatis, P.: Verifiable resource accounting for cloud computing services. In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security, CCSW'11, Chicago, pp. 21–26. ACM, New York (2011). doi:10.1145/2046660.2046666
59. Somorovsky, J., Heiderich, M., Jensen, M., Schwenk, J., Gruschka, N., Lo Iacono, L.: All your clouds are belong to us: security analysis of cloud management interfaces. In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security, CCSW'11, Chicago, pp. 3–14. ACM, New York (2011). doi:10.1145/2046660.2046664
60. Song, Z., Molina, J., Lee, S., Lee, H., Kotani, S., Masuoka, R.: Trustcube: an infrastructure that builds trust in client. In: Proceedings of the 1st International Conference Future of Trust in Computing, Berlin, pp. 68–79. Vieweg+Teubner (2009). doi:10.1007/978-3-8348-9324-6_8
61. symantec.com: W32.Fanbot.A@mm. http://goo.gl/NkX5h
62. Szeliski, R.: Image alignment and stitching: a tutorial. Found. Trends Comput. Graph. Vis. **2**(1), 1–104 (2006). doi:10.1561/0600000009
63. Thuente, D.J., Acharya, M.: Intelligent jamming in wireless networks with applications to 802.11b and other networks. In: Proceedings of the 2006 IEEE Conference on Military Communications, MILCOM'06, Washington, DC, pp. 1075–1081. IEEE Press, Piscataway (2006)
64. Verbelen, T., Simoens, P., De Turck, F., Dhoedt, B.: Cloudlets: bringing the cloud to the mobile user. In: Proceedings of the 3rd ACM Workshop on Mobile Cloud Computing and Services, MCS'12, Low Wood Bay, pp. 29–36. ACM, New York (2012). doi:10.1145/2307849.2307858
65. Walls, R.J., Learned-Miller, E., Levine, B.N.: Forensic triage for mobile phones with DEC0DE. In: Proceedings of the 20th USENIX Conference on Security, SEC'11, San Francisco, pp. 1–14. USENIX Association, Berkeley (2011)
66. Wei, J., Zhang, X., Ammons, G., Bala, V., Ning, P.: Managing security of virtual machine images in a cloud environment. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, CCSW'09, Chicago, pp. 91–96. ACM, New York (2009). doi:10.1145/1655008.1655021
67. Wilhelm, M., Martinovic, I., Schmitt, J.B., Lenders, V.: Short paper: reactive jamming in wireless networks: how realistic is the threat? In: Proceedings of the 4th ACM Conference on Wireless Network Security, WiSec'11, Hamburg, pp. 47–52. ACM, New York (2011). doi:10.1145/1998412.1998422
68. Xu, W., Ma, K., Trappe, W., Zhang, Y.: Jamming sensor networks: attack and defense strategies. Netw. Mag. Glob. Internetwkg. **20**(3), 41–47 (2006). doi:10.1109/MNET.2006.1637931
69. Zhang, X., Schiffman, J., Gibbs, S., Kunjithapatham, A., Jeong, S.: Securing elastic applications on mobile devices for cloud computing. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, CCSW'09, Chicago, pp. 127–134. ACM, New York (2009). doi:10.1145/1655008.1655026
70. Zhang, K., Zhou, X., Chen, Y., Wang, X., Ruan, Y.: Sedic: privacy-aware data intensive computing on hybrid clouds. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS'11, Chicago, pp. 515–526. ACM, New York (2011). doi:10.1145/2046707.2046767