Software Piracy Control Framework in Mobile Cloud Computing Systems

Mazhar Ali, Muhammad Usman Shahid Khan, Assad Abbas, and Samee U. Khan

{mazhar.ali, ushahid.khan, assad.abbas, samee.khan}@ndsu.edu

North Dakota State University, Fargo, ND, USA.

Keywords: Mobile Cloud Computing, Mobile Application, Software License, Piracy.

1. Introduction

The ubiquity of the cloud computing allows the mobile devices to connect and use the traditional cloud computing services [1]. However, unlike the normal computing machines, the mobile devices are resource constrained. The precincts of low processing power, less storage capacity, limited energy, and the capricious internet connectivity does not allow the compute and storage mandating applications to run on mobile devices [2]. The aforementioned limitations served as the motivation for a new computing paradigm called Mobile Cloud Computing (MCC) that enhances the abilities of mobile devices by offloading resource intensive applications to the cloud. Mobile devices can now execute heavy compute and storage intensive applications by using the computation and storage services of the cloud [3]. The MCC paradigm enables the users to access and manage their applications and data through mobile devices without the need to move to traditional computing machines.

The MCC possesses multifaceted advantages including **(a)** the ability to empower the mobile devices to perform compute intensive tasks, **(b)** boosting the mobile device performance by exploiting cloud computational capabilities, and **(c)** cutting down the energy consumption of mobile devices by delegating the computational tasks to cloud [4]. However, to benefit from the aforementioned capabilities of the MCC, specialized applications that support the MCC paradigm are required [5]. The reason for the development of the specialized applications is the inability of the traditional mobile applications to use the said characteristics of the MCC. To provide the above mentioned features, mobile application development models are used that offload the mobile applications or the smartphone clone to the cloud [2]. Nevertheless, the offloading of clone or application to the cloud may raise the software piracy issues [5]. The traditional mobile applications bind the application license mostly with the mobile device. In the MCC paradigm, when an application or the smartphone clone is offloaded to the cloud, the

device dependent parameters are no longer visible to the cloud. The absence of both the device parameters and the environment makes the cloud unable to verify the application license [5].

The issue of software piracy in mobile applications needs an immediate attention of the industry and academia due to the fact that the use of mobile devices is escalating with the swift rates. The survey conducted by the International Data Corporation (IDC) [6], states that by year 2015 the number of mobile devices requiring access to the Internet will exceed the number of conventional computers. Likewise, the number of cloud supported devices is expected to be around one trillion by the end of year 2014 [7]. Moreover, the development of mobile applications is anticipated to outstrip the development of conventional applications within next five years [8]. As a result of exponential growth in the development and usage of mobile devices and applications, the MCC market is estimated to outdo the value of $10 billion in year 2015 [7]. However, the aforesaid statistics also bring the issue of application piracy upfront with a severity level that never existed before.

According to the available statistics, Football Manager is the most pirated smartphone application. The Football Manager has 500 times more pirated copies than the legal copies [9]. Moreover, around 84% of the available iOS applications are affected by the piracy issues [9]. China that makes more than 25% of the total world's population has around 40% of pirated Android applications [10]. The piracy issues not only entail a financial loss to the application developers but may also increase the cost of cloud based services due to free usage. The pirated cloud application may utilize cloud services that can cost financial liabilities to the application providers. The year 2010 has observed 75% application downloads without payment that deprived application developers of 70% of the expected income [8]. Similarly, a research project has shown that the cloud enabled mobile applications can be run on cloud for free by manipulating the loopholes of the MCC applications [11]. The result of such happenings is a huge financial loss to not only the application developers but to the cloud service providers as well.

As mentioned earlier, the prevailing piracy control strategies do not solve the piracy issue in the MCC paradigm. Moreover, very little attention is paid to the piracy issue in the MCC by research community [5]. Therefore, it is the high time for the research community to focus on the piracy issues prevalent in MCC to cater the problem that not only involves financial concerns but also has an ethical obligations. The aforesaid situation demands the research efforts to develop a piracy control framework for mobile cloud applications.

In this chapter, we propose a software piracy control framework for mobile cloud applications. The proposed framework considers the piracy control issue as one of the offshoots of access control. The software license ensures that the software is executed only by the party that is authorized or has been granted access to execute. In the same manner, the proposed framework gives access to the requesting user to execute the application on the cloud only if the user is authorized to do so. The license verification and access grant in the proposed framework is ticket-based where the credentials and parameters are verified by the possession of the valid ticket or otherwise. The ticket is issued for a specified period and the execution of the application after that time will require the acquisition of the new ticket.

2. Related work

There are numerous strategies and products for license management in conventional computing paradigm. However, the research on license management in MCC has little been explored. In the following text we will explain few of the works that are based either on MCC or related to the cloud computing paradigm.

Khan *et al*. [5] presented a framework called pirax for piracy control in MCC environment. The pirax framework averts the unauthorized execution of mobile applications not only on a mobile device but also on cloud. The pirax is a node-locked licensing methodology that binds the device dependent parameters to enforce the license. On the device side, the pirax uses International Mobile Equipment Identity (IMEI) to bond the application with the mobile device. The application provider generates a unique ID for the application, and concatenates IMEI of requesting user and the generated application ID. Subsequently, the concatenated result is passed through a hash function. The resulting hash code is termed as a license that is applicable only at the mobile device. The license is transmitted to the mobile device after encrypting with the public key of the mobile device. The license is validated before application execution by generating the hash of application ID and IMEI locally and comparing with the received license. The successful comparison results in execution of the application on mobile device. However, when application is executed on the cloud, the IMEI cannot be accessed because the smartphone clone is offloaded to the cloud. Therefore, a request to the application provider is sent for providing cloud license for the application. The unique ID of the Virtual Machine (VM) that is assigned to the smartphone clone is also transmitted to the application provider. The application provider concatenates unique VM-ID, application ID, and previously generated license for mobile device. The result of the concatenation is passed through hash function to generate the cloud side license for the application. The same parameters are used at the cloud side to generate

hash for validating the license received from the application provider. However, the pirax framework works only for those mobile application models that require the whole smartphone clone to be offloaded to the cloud. Other models that require only parts of the application or mobile device are not handled by the pirax framework. Moreover, at the mobile device side IMEI is used that can be spoofed and the pirax does not deal with such hostile scenario.

The authors in [12] have presented GenLM that is token based approach for license management in grid and cloud environment. GenLM does not bind the license to a node or user. The license is attached to the data files that need to go for computations on the remote site. The user calculates the hash of the input data files. The hash of each of the input file is stored in a separate file that is termed as request token. The request token also contains the terms that are requested by the user from the application provider. Afterwards, the token is signed by the user with X.509 certificate. The signed request token is sent to the GenLM server. The GenLM server entertains the service requests for all of the application providers for license generation and verification. The server verifies the signature on the request token and forwards the request to the policy plugin of the application provider from which the license is requested. The policy plugin evaluates the user request on the basis of application provider's business model and decides whether to grant license on the requested terms or not. The policy plugin also handles the billing matters of the license. If the license is granted, the GenLM server signs the request token with its X.509 certificate. After signature of the GenLM server, the request token becomes the license token that is sent to the user. The user submits the license token along with the input data files to the grid or the cloud. The job is only executed if the hash of the input files in the license token matches the hash of the input files generated at the compute site. Although the proposed approach handles the computations at the grid and cloud, it only caters the computational data. Moreover, the technique is not valid for mobile cloud applications because the technique computes hash over the data only.

The pirax framework that we highlighted in this section is the only proposed framework that tries to handle the piracy issue in the MCC paradigm. However, the pirax framework has its own limitations that we discussed in the preceding discussion. In the next section, we present our proposed software piracy control framework for the mobile cloud computing environment. The proposed framework is ticket-based and is applicable to all mobile application models.

3. The proposed framework

The conventional node-locked phenomenon for enforcing piracy control does not prove to be fruitful in the mobile cloud computing environment. The reason for the aforesaid fact is that the device dependent parameters cannot be effectively accessed and employed in the cloud. The offloading to the cloud is done only for application, part(s) of application, or the whole clone of the mobile device. All of the aforesaid models cannot access and utilize the device parameters in the same way as on the device itself. Therefore, in the proposed framework we use a ticket-based mechanism to validate the license and grant access for the execution of the mobile application in the cloud.

3.1 Entities

The following entities are involved in the proposed methodology.

User: The clients that use the mobile application are the users in the proposed methodology. The users will purchase the mobile application from the application provider. The users can execute the mobile application either at the mobile device or at cloud at different points of time. The license needs to be enforced at both of the sides.

Application provider: The application provider is the developer of the mobile application. The application provider's objective is to eliminate the pirated and illicit execution of the mobile applications. The mobile application needs to be executed only by the customer who purchased and has rights for execution. The business model of the application provider can only be successful and fruitful by returning what is due for the application provider. A License Granting and Verification Server (LGVS) at the application provider's end handles the users' request for granting license and execute permissions in the form of a ticket. The application either at device or cloud needs a ticket to verify the genuineness of the user license for subsequent execution.

The Cloud: The cloud provides computational services to the users. The users offload their application to the cloud to take advantage of the computational power of the cloud. Moreover, the performance and energy consumption of the mobile device is enhanced by outsourcing the computations to the cloud. The execution of the offloaded mobile applications to the cloud is managed by Access Granting Server (AGS) that is present at the cloud. The AGS at the cloud and the LGVS at the application provider's site work in coordination to control the illicit mobile application's execution.

3.2 The proposed methodology

In this section, we present the details of the proposed framework to prevent the illicit execution of mobile applications. The framework ensures that the application is executed only by the authorized user. Initially, the user contacts the application provider to purchase the application. The pricing and the license terms are agreed upon according to the procedure set by the application provider. Subsequently, the user installs the application on the mobile device. However, the execution of the mobile application needs the permission of the application provider. The same permission is needed to execute the application after offloading to the cloud.

When the application needs to be executed on the mobile device, the first time execution is interrupted to acquire the ticket from the LGVS. The application extracts the IMEI of the mobile device. The IMEI is sent to LGVS along with the application ID, and the requesting user ID. The IMEI is encrypted with the public key of LGVS and subsequently signed by the private key of the mobile device. The encryption and signing are meant to prevent IMEI spoofing by unauthorized users and applications. The LGVS authenticates the identity of the user and application and checks whether the user has obtained a valid license or not. A locally maintained database contains the information about the issued licenses and user identities. After successful authentication, the LGVS prepares and sends the following message to the application.

$$\text{Tic\_MD} = \{\{\{IMEI||Application\ ID||validity\ period\}_{pub\_MD}\}_{pri\_LGVS}\}$$

where *Tic_MD* represents the license for application execution on mobile device,
|| represents the concatenation operator,
pub_MD = public key of the mobile device.
pri_LGVS = private key of LGVS
The application at the mobile device verifies the LGVS signatures, extracts the application ID and IMEI and compares them with local values of the same parameters. The validity period is verified and upon success the mobile application is allowed to execute. At the mobile device, the strategy is more like a node-locked phenomenon. However, at cloud side the process is different. At the device end, the *Tic_MD* can be stored by the application and can be used during the lifecycle of the license. However, to execute application at the cloud, the license is verified every time the application is executed.

Whenever a user wants to execute the mobile application at the cloud or at the device, he/she contacts the LGVS of the corresponding application provider. The user sends authentication credentials and the application ID for which the user has obtained the license. The LGVS after

successful authentication checks whether the user has got a valid license or not. The check is performed from a locally maintained database at LGVS. In case of a valid license, the LGVS sends a message to the mobile user with the following contents:

$$M = \{\{nonce\}_{pub\_user}, \{AT\}_{pub\_AGS}\}$$

where AT = Access Token,
pub_user = public key of the user,
pub_AGS = public key of the Access Granting Server,
$\{nonce\}_{pub\_user}$ = nonce encrypted with the public key of the user,
$\{AT\}_{pub\_AGS}$ = AT encrypted with the public key of AGS.
The AT is further comprised of the following attributes.

- Requesting user ID
- Token validity period
- Application ID
- Nonce

The mobile user decrypts the nonce. The AT cannot be decrypted by the mobile user as it is encrypted by the public key of AGS. To execute the application at the cloud, the mobile user has to access the cloud through the AGS. The mobile user prepares the following message to send to the AGS along with the authentication credentials.

$$M_{AGS} = \{\{\{nonce\}_{pub\_AGS}\}_{pri\_user}, \{AT\}_{pub\_AGS}\}$$

In the above given message, the nonce is signed by the sending user after encrypting with the public key of the AGS. The AGS verifies the user signature and decrypts the nonce. Subsequently, it compares the nonce received by the mobile user and the nonce received from LGVS. The aforesaid step authenticates the requesting user. Moreover, the AGS verifies the validity of the license by extracting parameters from the AT. The AGS verifies the application ID received in the AT by comparing with the application ID of the offloaded application. The validity period of the token is also verified. If the license is valid, the AGS grants access to the user for executing the mobile application at the cloud. The request is denied otherwise. Figure 1 shows the general architecture of the proposed framework while Figure 2 depicts the workflow of the proposed framework for executing the application at the cloud.

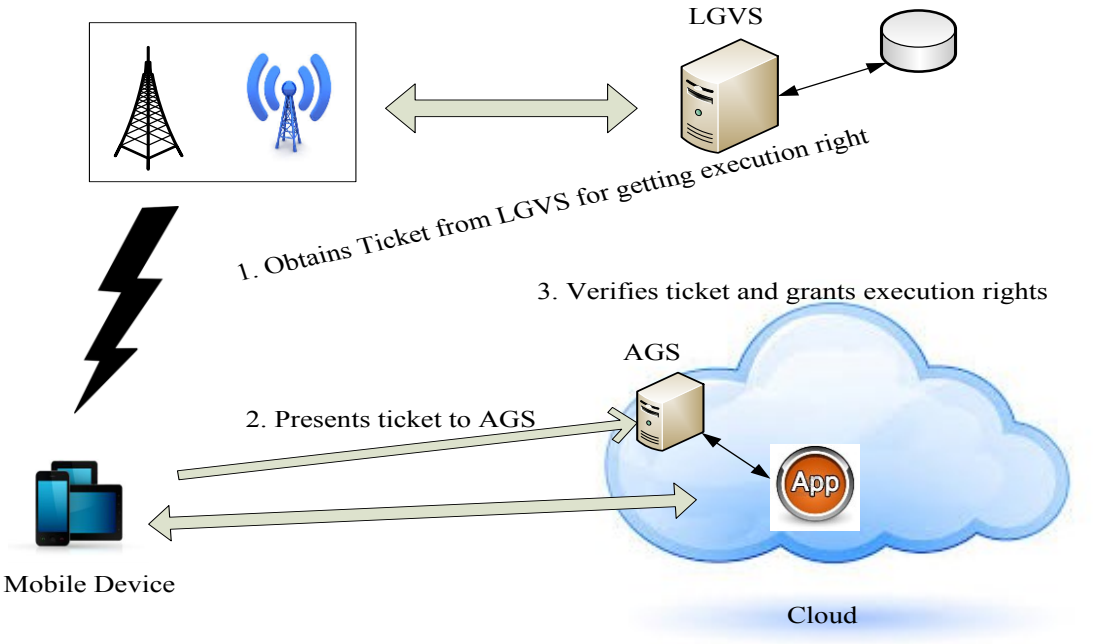

Fig. 1. General architecture for proposed framework

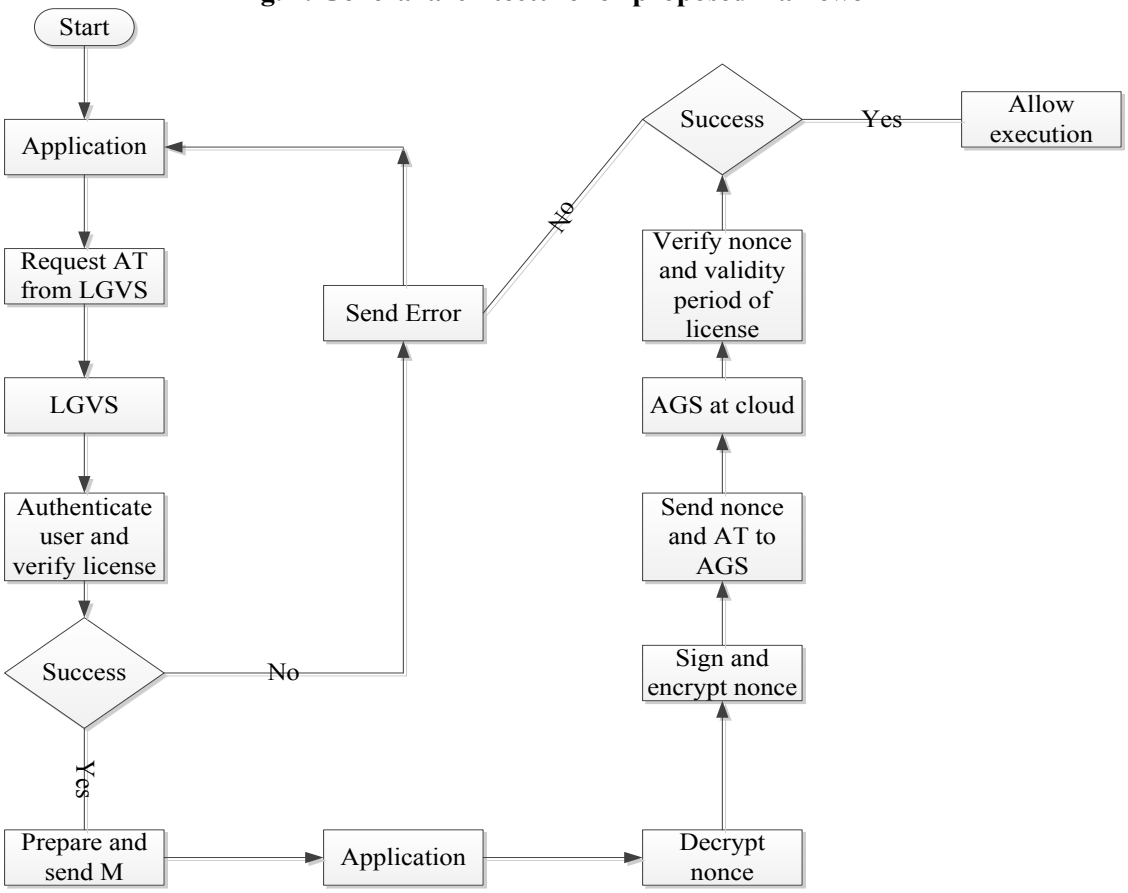

Fig. 2. Workflow of proposed framework for mobile application execution at cloud

4. Discussion

The proposed framework has two courses of actions to deal with the application execution at the mobile device and at the cloud. The process adopted at the device end is much like a node-locked phenomenon. However, at the cloud side the ticket needs to be issued by the LGVS and verified by the AGS each time the application is offloaded to the cloud. It is noteworthy that the proposed framework is flexible enough to work with all of the models for application offloading to the cloud. The framework works with the complete clone or only application or part(s) of application offloaded to the cloud. The proposed framework operates on the basis of ticket issuance. The ticket is issued by the application provider and verified by the AGS at the cloud. The validity period and the users are authenticated each time the request for application execution is initiated.

The licenses are generated by the LGVS that is proposed to be managed by the application provider. The communication of vital parameters is secured by use of public/private keys that can be managed by X509 certificates. Therefore, the proposed framework also considers the security aspect to provide better services to both the application providers and the users.

5. Conclusions

The mobile applications are offloaded to cloud to enhance the mobile device's performance and take advantage of the cloud's computational capabilities. However, offloading the mobile application to the cloud raises application piracy concerns. We proposed a software piracy control framework in mobile cloud computing systems. The framework approaches the issue from the access control perspective and is based on tickets for verification of a valid license. A License Granting and Verification Server (LGVS) verifies the user and grants a ticket for certain time. The Access Granting Server (AGS) after verification of the ticket issued by LGVS, grants execution rights to mobile application at the cloud. The cryptographic parameters are used in the proposed framework to obtain the verification services and keep the messaging secure.

References

[1] A. N. Khan, M. L. M. Kiah, S. U. Khan, and S. A. Madani, "Towards secure mobile cloud computing: A survey," *Future Generation Computer Systems*, Vol. 29, No. 5, 2013, pp. 1278-1299.

[2] A. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," IEEE Communications Surveys & Tutorials, Vol. 16 , No.1, 2013, pp. 393-413.

[3] A. N. Khan, M. L. M. Kiah, M. Ali, S. A. Madani, and S. Shamshirband, "BSS: block-based sharing scheme for secure data storage services in mobile cloud environment," *The Journal of Supercomputing*, Vol 70, No. 2, 2014, pp. 946-976.

[4] A. N. Khan, M. L. M. Kiah, S. U. Khan, and S. A. Madani, "A study of incremental cryptography for security schemes in mobile cloud computing environments," In *2013 IEEE Symposium on Wireless Technology and Applications (ISWTA),* 2013, pp. 62-67.

[5] A. R. Khan, M. Othman, M. Ali, AN Khan, and S. A. Madani, "Pirax: framework for application piracy control in mobile cloud environment," *The Journal of Supercomputing*, Vol. 68, No. 2, 2014, pp. 753-776.

[6] International Data Corporation, http://www.idc.com/getdoc.jsp?containerId=prUS23028711, Nov. 2014.

[7] Preston A. Cox, "Mobile cloud computing - Devices, trends, issues, and the enabling technologies," available at http://public.dhe.ibm.com/software/dw/cloud/library/cl-mobilecloudcomputing-pdf.pdf, Nov. 2014

[8]Worldwide Application Development and Deployment http://www.idc.com/getdoc.jsp?containerId=IDC_P1600, Nov. 2014

[9] Dear Android: a 9:1 piracy rate for games is not good enough http://www.wired.co.uk/news/archive/2012-05/02/androidmarket-game-piracy. Nov. 2014

[10] S. Hanna, L. Huang, E. Wu, S. Li, C. Chen, and D. Song, "Juxtapp: A scalable system for detecting code reuse among android applications," *In Detection of Intrusions and Malware, and Vulnerability Assessment, Springer Berlin Heidelberg*, 2013, pp. 62-81.

[11] V. Tendulkar, R. Snyder, J. Pletcher, K. Butler, A. Shashidharan, and W. Enck, "Abusing cloud-based browsers for fun and profit," In Proceedings of the 28th Annual Computer Security Applications Conference, 2012, pp. 219-228.

[12] M. Dalheimer and F-J. Pfreundt, "Genlm: license management for grid and cloud computing environments," In *9th IEEE/ACM International Symposium on Cluster Computing and the Grid,* 2009, pp. 132-139.