2012 Tenth International Conference on ICT and Knowledge Engineering

A Proposal to Improve the Security of Mobile Banking Applications

Mahmoud Elkhodr[1], Seyed Shahrestani[1] and Khaled Kourouche[2]
School of Computing, Engineering and Mathematics[1]
School of Business[2]
University of Western Sydney
Sydney, Australia

***Abstract*— Mobile banking (m-banking) is considered to be one of the most important mobile commerce applications currently available. The ubiquitous access to data with no place restrictions helps to promote this technology. The security and privacy of sensitive financial data is one of the main concerns in acceptance of these systems in Australia. It is specifically important to secure the transmission of the financial data between the financial institutions' server and the mobile device used by consumers, as their communications are via unsecured networks such as the Internet. In this paper, a trust negotiation approach is proposed to address these security concerns. Trust negotiation is combined with the Transport Layer Security (TLS) as the underlying protocol. This combination of technology aims to maximize the existing security of m-banking applications. It results in significant improvements in security compared to the traditional identity-based only access control techniques. The proposed approach is implemented as a mobile application. It demonstrates that the developed application is easy to use and deploy in typical mobile environments.**

***Keywords*— Security, Mobile Computing, Mobile Banking, Ubiquitous Access**

I. INTRODUCTION

The advancement of mobile technology has revolutionized the way people use mobile devices in their everyday lives. Nowadays, Mobile phones are not only used for communication purposes i.e. making and receiving phone calls or sending text messages, these Mobile phones, commonly referred to as 'smart' phones, are used as an information distribution platform and often as computing devices. In particular, the rapid development in mobile commerce applications has reshaped traditional banking services to form a mobile banking (m-banking) industry. New banking services are now available which enable a mobile client to request and receive information about their personal account, to transfer funds between accounts and make electronic bill payments anywhere and at anytime by simply using applications installed on their mobile devices. M-banking applications are increasingly becoming more attractive to consumers and in some cases, may even be preferred over e-banking. The availability of access and no place restriction, as shown in Fig. 1, are among the main factors that help in saving a customer time and reducing their expenses. This telecom market is one of the largest and the fastest growing markets given low costs, increasing computational power, and ease of use. Significantly, mobile cellular subscriptions reached approximately 6 billion worldwide in 2011 [1]. These figures will lead to an increase in mobile application usage. However, given that mobile devices may be vulnerable to threats, attack and loss; security is a major area of concern.

In this context, ensuring secure access to financial data (FD) through a heterogeneous mobile device or tablet and adapting the information to the properties of this device, in a secure way, are two problems raised and remain unresolved. Other significant aspects also need to be considered, such as content display, memory constraints and the ability to incorporate security features; which add more complexity to the issue of adaptation. In this work, our concern is to secure access to the FD through a mobile device and during transmission where sensitive data are more likely to be exchanged. We propose a security approach for m-banking; and a prototype solution that aims to secure the transmission and access to personal and financial data over wireless and 3G networks through mobile devices. This approach authenticates the requester, the device in use, and secures the communication channel as shown in Fig. 2. The rest of this paper is structured as follows: some related work on m-banking is given in Section II. The proposed secure mobile banking approach (SMBA) is presented in Section III. Section IV reports the experiment and evaluation of results. Section V concludes this paper.

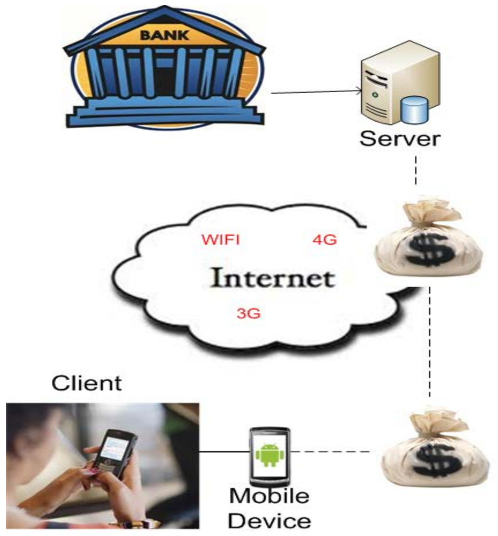

Figure 1. Mobile Banking Applications

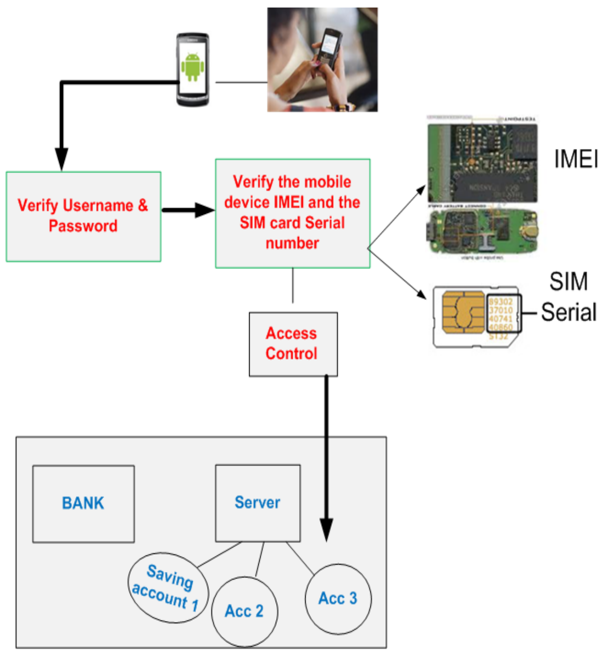
Verify Username & Password
Verify the mobile device IMEI and the SIM card Serial number
IMEI
SIM Serial
Access Control
BANK
Server
Saving account 1
Acc 2
Acc 3

Figure 2. The Proposed Secure Mobile Banking Approach

II. MOBILE BANKING

M-banking is considered to be one of the most value-added and important mobile commerce applications currently available [2]. Mobile commerce has several definitions in the literature but can be generally defined as a type of e-commerce transaction, conducted through mobile devices using wireless telecommunications networks and other wired e-commerce technologies [3]. The main success factors of mobile commerce are its convenience, ease of use, ubiquity and trust [4]. Mobile commerce applications can be accessed anywhere and at anytime and is enabled through different technologies such as networking, embedded systems, database and security [5]. Mobile hardware, software and wireless technology enable mobile commerce applications which can transmit data quickly and locate a user position.

The key players in m-banking include banks and other financial institutions such as credit card companies, mobile operators and retailers [6]. Their success in increasing m-banking uptake amongst clients will ultimately rely upon how secure the FD and transactions are. That is, the end user must be confident in the financial institution carrying out the transaction, the transmission of FD and the technology itself. Although the literature has attempted to address the issue of security, the literature is limited and suffers from shortfalls.

In [7], an M-Payment architecture which links up an operator, service provider and financial institution has been proposed. It allows mobile consumers to purchase goods/services using SMS and WAP. While this solution forms an alternative payment option, it does not provide sufficient study on the service delivery. In [8], the model proposes the use of a biometric authentication mechanism. The system software would be installed onto a device that has a supporting finger print scanner. The finger print template would be captured on the phone and compared against a stored template on a database server. However, approaches like these will require the client to carry an additional piece of hardware, i.e. the finger print scanner. Other studies, such as in [9] suggested a method for secure payment services through mobile phones by using Bluetooth technology from modern information system technology. While the security of Bluetooth technology systems is higher than Internet-based systems, there are known limitations related to Bluetooth. Nevertheless, a recent study on m-banking application identified the security requirement to be as follows: data needs to be encrypted, access to the data must be authorized and the authorization has to be simple [9].

The issue of security in m-banking becomes much clearer when one considers the dramatic rise in m-banking transactions globally. In a survey conducted by KPMG [10] covering 5,627 consumers in 22 countries, it was found that in the United States, about 30 percent of households use their mobile phones to perform banking operations while in Asia (India, China and Korea) it is higher at 43 percent of households. Interestingly, only 19 percent of households in Australia use m-banking applications. This finding for Australia is not due to a lack of mobile phone subscribers. In actual fact, Australia has one of the highest levels of subscriber penetration rates at about 125% of the population [10], i.e. there are around six million more mobile phone subscribers than people. However, it can be attributed to Australians having a lower level of comfort in using their mobile phone for financial transactions. Twenty one percent of Australians are comfortable with m-banking compared to 40 percent in Asia and 34 percent globally. It is clear that mobile phone users have concerns over security and privacy with 59 percent of Australian respondents sharing this concern [10].

Significantly, in addressing the security concerns of mobile clients surrounding m-banking, financial institutions can cut down the cost of providing banking services to customers. For example, Infogile Technologies estimates that an average teller or phone transaction costs about $2.36 each, whereas an electronic transaction costs only about $0.10 each [11]. Additionally, this new channel gives the bank ability to cross-sell or up-sell their other complex banking products and services such as vehicle loans, credit cards etc. Yet another benefit is the anywhere/anytime characteristics of mobile services. A mobile phone is almost always with the customer. As such it can be used over a vast geographical area. The customer does not have to visit the bank ATM or a branch to avail of the bank's services.

III. THE SECURE MOBILE BANKING APPROACH

For securing the transmission of clients' FD, the Transport layer Security (TLS) protocol is combined with a proposed trust negotiation method. This method authenticates the client, the mobile device used in accessing the bank account information, and the server where FD are stored. This combination of technology, referred to as the secure mobile banking approach (SMBA), guarantees the confidentiality and the secure disclosure of clients' FD.

SMBA's operation starts by establishing a secure session between the client's mobile device and the bank's server. The bank server is where the client's FD are actually stored. This session is created using the TLS handshake mechanism. It is used to ensure the encryption of the exchanged messages and the protection against intruders. After establishing the secure session, SMBA executes the authentication method. It first starts by authenticating the requester (the client), using a username and password, to the bank's server. This process is illustrated in Fig. 3.

The second level of authentication, the device in use authentication, starts after identifying the client. This process runs silently in the background without the user interference. It is made transparent to the users because they are considered to be non-expert users of technology. Therefore, by running this authentication mechanism in the background, we aim at minimizing the user's inputs and optimizing the security of the system by providing extra protective access control features. Authenticating the device in use requires the exchange of digital credentials related to the device itself. These are the attributes which allows a particular device to be identified among others. The International Mobile Equipment Identity (IMEI) and the subscriber identification module (SIM) serial numbers are the credentials used to identify the mobile device used by the requester. IMEI is a unique number used for identifying mobile devices. It is only used for identifying the device and has no permanent or semi-permanent relation to the user. A SIM number is burned on the integrated circuit of the actual SIM card that securely stores the service-subscriber key (IMSI). This number is used to identify a subscriber on mobile telephony devices. The combination of the IMEI and SIM serial numbers will lead into the verification of the mobile device and the SIM card inserted. This will ultimately result in authenticating the device in use. Upon the successful identification of the client and the mobile device used, access to particular FD can then be granted.

For the successful operation of the SMBA, some conditions need to be met. Therefore, before deploying the application and putting it into work, the following requirements must first be achieved:

- The client should be known to the bank system and he/she has been assigned a Username and Password. This is an obvious requirement necessary for authenticating the requester.
- The clients' mobile device should also be known to the bank server. The system should hold a record of the clients' mobile IMEI and the SIM card details. These are the digital credentials used for identifying the device in use.

After the creation of a TLS session, access to FD will be secured using the proposed approach. Therefore, this process will ensure that data were only disclosed to the authorized person, using an authorized device.

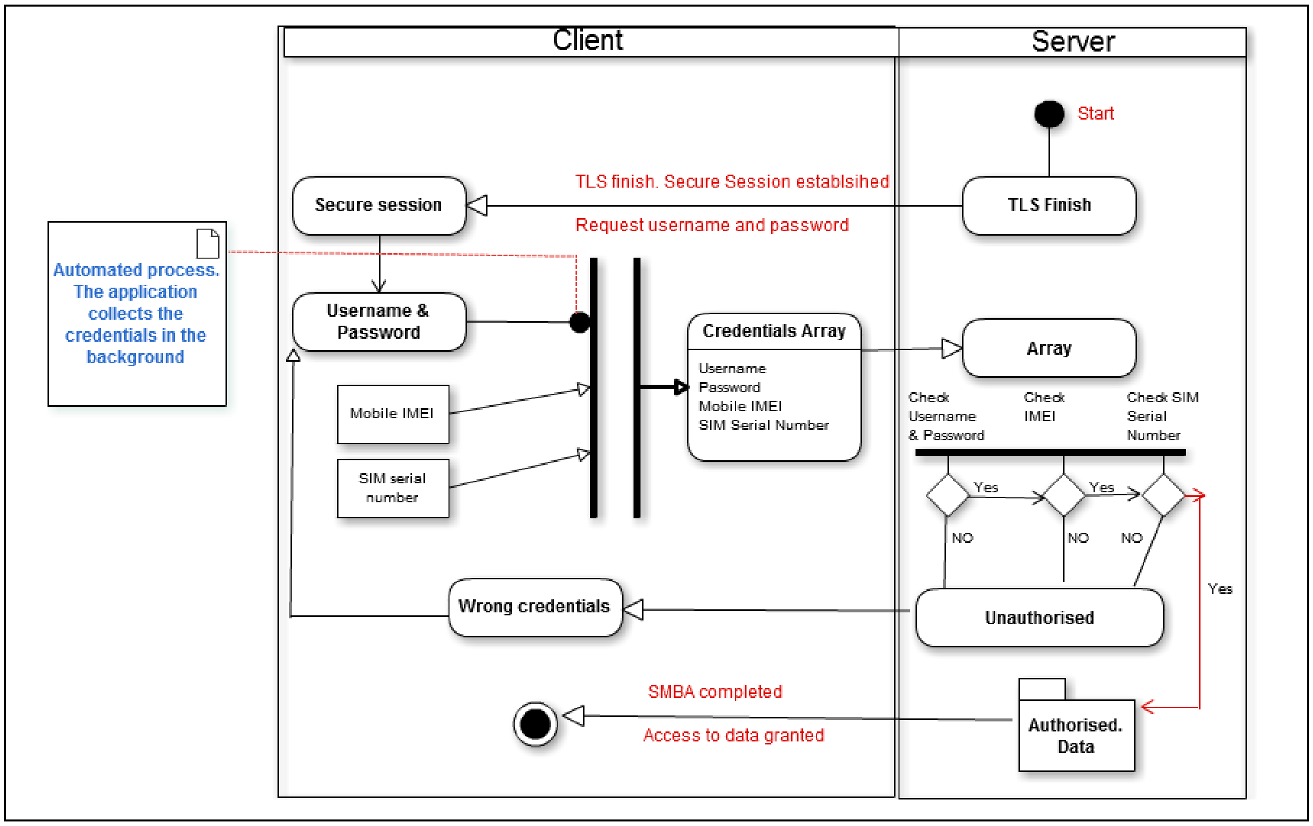

Figure 3. The SMBA Acitvity Diagram

IV. THE MOBILE APPLICATION AND ITS DEVELOPMENT

The chosen platform for simulating and developing the mobile application is Android. Since Android is an open development platform providing access to the device hardware, ports, background services, notifications and others are open and free to use. Hence, it is considered a suitable platform for research and experiment. Android is based upon a modified version of the Linux kernel. It is a software stack for mobile devices that includes an operating system, middleware and key applications. A wide range of mobile devices and tablets, known as Android pad, run Android as an operating system. We name Samsung Galaxy SII, HTC Desire, Samsung Galaxy and many others. For constructing and testing the mobile application, we developed a simulation using the Android SDK. Android SDK includes a virtual mobile device. It is an emulator for developing and testing Android applications without using a physical mobile device. Before applying the proposed SMBA method, simulating access to a bank's personal account by a client is required. The application was constructed in two parts: The first part is a client side application which has a graphical user interface allowing clients to enter their credentials and request access to their accounts. The second part is a server side application which holds the FD stored in a database. It is responsible for generating the server's responses, the requests and the execution of the proposed SMBA methods.

The client side application was developed using the Eclipse framework. Eclipse is popularly known as a Java IDE. The following packages necessary for the development of the application were added and configured in Eclipse:

1- Android Dalvik Debug Monitor Server (DDMS) and Android Development Tools (ADT): DDMS is used as a middleware connecting Eclipse to the applications running on the device. Also, DDMS is needed because every application on Android runs in its own process; known in java as a thread. DDMS will manage the multi-threading process. Therefore, it hosts the application on its own virtual machine. ADT is used to create the application's user interface. It uses a custom Extensible Markup Language (XML) for that purpose.

2- Android Documentation version 2.2, the Application Programming Interface version 8 (API 8), revision 1: Used for debugging the application.

3- The Android software development kit version 2.2 (SDK Platform Android 2.2), API 8, revision 2: Interaction between the application and the Android system is done through the API version 8.

This application uses the protocol TLS version 1.0 for securing the communication between its browser (Web HTTP) and the web server (Apache). For the purpose of initiating a secure session, a custom Java interface called *MyHttpClient* was implemented within the class login.java. This interface uses an object for creating HTTPS connections. This object encapsulates the self-signed certificate and the server key; which are necessary for the establishment of TLS sessions. It operates over the port number 443. Selected code of the class login.java is given in Table I.

The Object *MyHttpClient*, created in class login, is responsible for communicating the client to the server. It generates the client messages used for establishing the handshake via the method *createClientConnectionManager*. The last message in Class *MyHttpClient*, return *new TLSSocketFactory(trusted),* is a method used to validate the web server. It also authenticates the certificate to the web server using a private key. The command *KeyStore trusted = KeyStore.getInstance("BKS") is used to store the file containing the trusted certificate.*

In case a non-trusted certificate is used, the client secure socket will reject the connection during the TLS session handshake. After establishing the secure session, the client side application collects the username and the password, from the client, using the application interface. It also collects the mobile device's IMEI and the SIM card serial numbers using the Telephony Service class and stores them into variables, as in the code shown below.

StringsimSerial=((TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE)).getSimSerialNumber();
StringIMEI=((TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE)).getDeviceId();

These credentials, collected from the mobile device, are added to the array "*nameValuePairs*" along with the username and password, previously collected. This array is then sent to the server over HTTPS using the *httpPost* method. The server subsequently proceeds with the verification process as described in the SMBA method given in Section III.

httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));

TABLE I. CODE EXTRACTED FROM CLASS LOGIN.JAVA

```
public MyHttpClient(Context context) {
this.context = context;  }
protected ClientConnectionManager
createClientConnectionManager() {
SchemeRegistry registry = new SchemeRegistry();
registry.register(new Scheme("http",
PlainSocketFactory.getSocketFactory(), 80));
registry.register(new Scheme("https",
newTLSSocketFactory(), 443));
return new; SingleClientConnManager(getParams(), registry);
}
private TLSSocketFactory newTLSSocketFactory() {
   try {
KeyStore trusted = KeyStore.getInstance("BKS");
     InputStream in =
context.getResources().openRawResource(R.raw.mystore);
     try createClientConnectionManager finally {
       in.close();
     }
return new TLSSocketFactory(trusted);
```

Upon the receipt of the client's message, the server side application starts verifying the credentials included in the array. The server side application consists mainly of the class login.php; which has the responsibility of comparing the received credentials with those stored in the database. However, for the server to be able to read, parse and respond to the clients' messages, a server API is needed. One of the main functionality of this API is to generate the server's responses. A server's response can be of two types: 1.) a server response with a status code saying "Unauthorized Login", or 2.) an authorized login with a correct status code and a body message. The body message is the bank account information requested. The server API consists of two main subclasses:

- RestUtils.php: This class is responsible for processing the incoming requests. It is also responsible for sending the appropriate responses to the client.
- RestRequest.php: The wrapper class that represents the server's request.

Therefore, using this API, the server is able to process the messages sent by the client. The client, on the other side, is specifically configured to send the array of credentials as a message to the class login.php, using the following statement (extracted from login.java class).

HttpPost httppost = new HttpPost("https://ServerAddress/hospital/login.php");

The class Login.php receives this message and using the server API; it reads and parses the credentials included. The process of reading and parsing the values from the array is made using the API's function *Processrequest()* implemented in class login.php as shown in the following codes:

include("rest.php");
$data = RestUtils::processRequest();
$requestVars = $data->getRequestVars();

Next, the values extracted from the array are stored in PHP variables, as shown in the code below. These are used to perform the verification process.

$USERNAME = $requestVars['username'];
$PASSWORD = $requestVars['password'];
$SERIAL = $requestVars['serial'];
$IMEI = $requestVars['imei'];

The server proceeds into the SMBA verification process by verifying the client's username and password, stored in the "$USERNAME" and "$PASSWORD" PHP's variables. It queries the database to verify if these values exist, as shown in the following statement:

$checkUserQuery = "SELECT id FROM Clients WHERE username='" . $USERNAME . "' AND password='" . $PASSWORD . "'";
$checkUserExecute = mysql_query($checkUserQuery);
$userRow = mysql_fetch_array($checkUserExecute);

If the server failed to verify the username and password, it generates an unauthorized message using the following statement:

RestUtils::sendResponse(401, '', 'application/json');

This instructs the server's API to send an unauthorized message to the client. The server then ends the negotiation process and waits for further actions from the client; or the session expires. If the verification succeeded, the server has successfully authenticated the client. Next, the server proceeds into verifying the device's in use credentials, the IMEI and the SIM serial numbers included in the array. To achieve this, the server extracts the client's identification number (ID) from the database. This ID will be used, by the server, to access his or her records on the database. Using this ID, the server is able to access the registered credentials. It compares the received IMEI and SIM serial number stored on the database, with those included in the array sent by the client. This process is done using the "Switch" function given in Table III.

If authentication succeeds, the server generates a status code numbered "200". However, if any of the credentials failed to authenticate, the server halts the verification process, using a break statement, and generates a different status code numbered "401". This tells the client that authentication has failed.

On the other side, the client side application is waiting for the server's response to act accordingly. Therefore, upon the receipt of the server's message, the client converts the status code into a readable message to the user. In case the status code number received was 200, the function calls the activity "*ViewAccount.java*", given in Table II. This activity displays to the requester, the client, the list of the authorized services, such as the ability to access bank account or paying bill as shown in Fig. 4.

TABLE II. ACTIVITYT VIEWACCOUNT.JAVA

```
if (!statusCode.equals("200"))
{
Toast toastUnauthorized =
Toast.makeText(getApplicationContext(), "Unauthorized
Login, please try again or exit the application",
Toast.LENGTH_SHORT);
toastUnauthorized.show();
}
Else
{
Intent i = new Intent(getApplicationContext(),
ViewAcconts.class);
i.putExtra("com.site90.experiment.Bankportal.username",
usernameString);
startActivity(i);
}
```

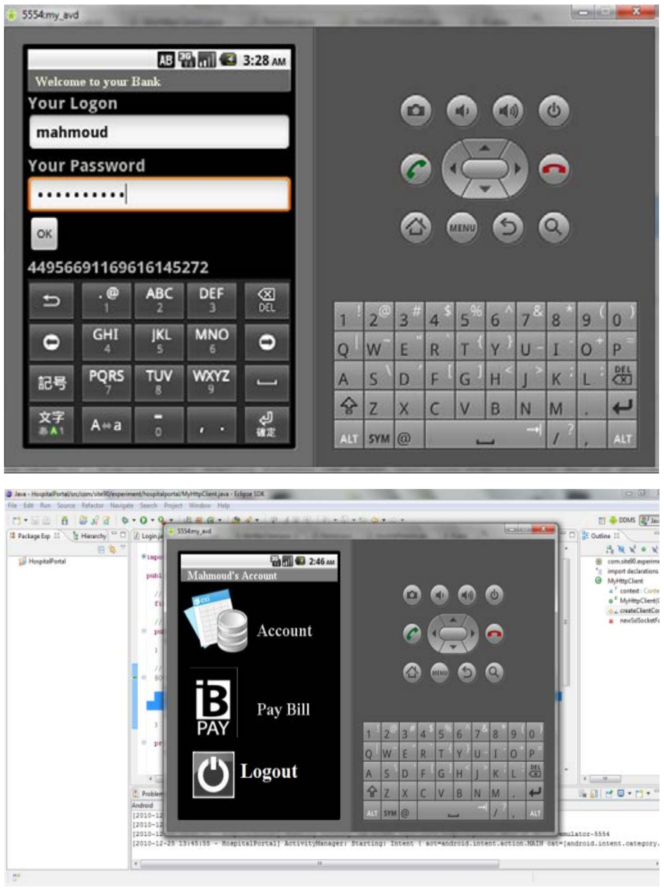

Figure 4. The Mobile Application Screenshots

V. CONCLUSION

This work proposed to provide financial institutions' clients with a mobile application that could be used to access their personal or business accounts anywhere and at anytime in a secure way. The proposed approach verifies the requesters and their device. Therefore, it gives users the opportunity of registering their mobile devices and also gives the financial institutions a way to verify the device in use. This approach substitutes and enhances the SMS two-factor authentication method by automating the authentication process. This clearly enhances the security of m-banking systems by adding extra protection features to the authentication mechanism. It also improves the user experience by minimizing the users' inputs. Although the mobile device emulator used for testing the proposed application mimics all the hardware and software features of a physical mobile device, some possible problems might arise when a real mobile is used. This and others issues such as, performance and delays are among the limitations that need to be considered. Our future work will address these issues as we are in the process of integrating this approach with a location verification method to the m-banking application.

REFERENCES

[1] "The World in 2011 — ICT Facts and Figures," International Telecommunication Union (ITU) 2011.

[2] Tzong, *et al.*, "Adoption of mobile Location Based Services with Zaltman Metaphor Elicitation Techniques," *Int. J. Mob. Commun.*, vol. 7, pp. 117-132, 2009.

[3] K. Siau, E.-P. Lim, and Z. Shen, "Mobile Commerce: Promises, Challenges and Research Agenda," ed: IGI Global, 2001, pp. 4-13.

[4] G. Xu and J. A. Gutierrez, "An Exploratory Study of Killer Applications and Critical Success Factors in M-Commerce," ed: IGI Global, 2006, pp. 63-79.

[5] U. Varshney and R. Vetter, "Mobile Commerce: Framework, Applications and Networking Support," *Journal on Mobile Networks and Applications,* vol. 7, pp. 185-198, 2002.

[6] N. Mallat, M. Rossi, and V. K. Tuunainen, "Mobile banking services," *Commun. ACM,* vol. 47, pp. 42-46, 2004.

[7] Y. Chou, C. Lee, and J. Chung, "Understanding m-commerce payment systems through the analytic hierarchy process," *Journal of Business Research,* vol. 57, pp. 1423-1430, 2004.

[8] M. Gordon and S. Sankaranarayanan, "Biometric security mechanism in Mobile paymentts," in *Wireless And Optical Communications Networks (WOCN), 2010 Seventh International Conference On,* 2010, pp. 1-6.

[9] K. Pousttchi and M. Schurig, "Assessment of today's mobile banking applications from the view of customer requirements," in *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on,* 2004, p. 10 pp.

[10] (2010), *Mobile banking rising globally, but Australia yet to fully embrace* Available: www.kpmg.com/convergence

[11] "Mobile Banking – The Future," Infogile Technologies 2007.