International Journal of Mathematics and
Computer Science, **4**(2009), no. 2, 65–80

Multi Factor Authentication Using Mobile Phones

Fadi Aloul[1], Syed Zahidi[1], Wasim El-Hajj[2]

[1]Department of Computer Science & Engineering
American University of Sharjah
United Arab Emirates

[2]College of Information Technology
United Arab Emirates University
United Arab Emirates

e-mail: faloul@aus.edu, b00017408@aus.edu, welhajj@uaeu.ac.ae

(Received March 25, 2009, Accepted July 22, 2009)

Abstract

This paper describes a method of implementing two factor authentication using mobile phones. The proposed method guarantees that authenticating to services, such as online banking or ATM machines, is done in a very secure manner. The proposed system involves using a mobile phone as a software token for One Time Password generation. The generated One Time Password is valid for only a short user-defined period of time and is generated by factors that are unique to both, the user and the mobile device itself. Additionally, an SMS-based mechanism is implemented as both a backup mechanism for retrieving the password and as a possible mean of synchronization. The proposed method has been implemented and tested. Initial results show the success of the proposed method.

1 Introduction

Today security concerns are on the rise in all areas such as banks, governmental applications, healthcare industry, military organization, educational institutions, etc. Government organizations are setting standards, passing laws and forcing organizations and agencies to comply with these standards

Key words and phrases: Authentication, Integrity, Mobile Security.
AMS (MOS) Subject Classification: 68M10.
ISSN 1814-0424 © 2009, http://ijmcs.future-in-tech.net

with non-compliance being met with wide-ranging consequences. There are several issues when it comes to security concerns in these numerous and varying industries with one common weak link being passwords. Most systems today rely on *static passwords* to verify the user's identity. However, such passwords come with major management security concerns. Users tend to use easy-to-guess passwords, use the same password in multiple accounts, write the passwords or store them on their machines, etc. Furthermore, hackers have the option of using many techniques to steal passwords such as shoulder surfing, snooping, sniffing, guessing, etc.

Several 'proper' strategies for using passwords have been proposed. Some of which are very difficult to use and others might not meet the company's security concerns. Two factor authentication using devices such as tokens and ATM cards has been proposed to solve the password problem and have shown to be difficult to hack. Two factor authentication also have disadvantages which include the cost of purchasing, issuing, and managing the tokens or cards. From the customer's point of view, using more than one two-factor authentication system requires carrying multiple tokens/cards which are likely to get lost or stolen.

Mobile phones have traditionally been regarded as a tool for making phone calls. But today, given the advances in hardware and software, mobile phones use have been expanded to send messages, check emails, store contacts, etc. Mobile connectivity options have also increased. After standard GSM connections, mobile phones now have infra-red, Bluetooth, 3G, and WLAN connectivity. Most of us, if not all of us, carry mobile phones for communication purpose. Several mobile banking services available take advantage of the improving capabilities of mobile devices. From being able to receive information on account balances in the form of SMS messages to using WAP and Java together with GPRS to allow fund transfers between accounts, stock trading, and confirmation of direct payments via the phone's microbrowser [12]. Installing both vendor-specific and third party applications allow mobile phones to provide expanded new services other than communication. Consequently, using the mobile phone as a *token* will make it easier for the customer to deal with multiple two factor authentication systems; in addition it will reduce the cost of manufacturing, distributing, and maintaining millions of tokens.

In this paper, we propose and develop a complete two factor authentication system using mobile phones instead of tokens or cards. The system consists of a server connected to a GSM modem and a mobile phone client running a J2ME application. Two modes of operation are available for the users based on their preference and constraints. The first is a stand-alone approach that is easy to use, secure, and cheap. The second approach is an SMS-based approach that is also easy to use and secure, but more expensive. The system has been implemented and tested. Initial results are very promising.

In the next section we provide a general background about authentication factors and existing two factor authentication systems. Section 3 describes the proposed system, the OTP algorithm, client, server, and the database. Section 4 provides results of testing the system. In section 5 we conclude the paper and provide future work.

2 Background

By definition, authentication is the use of one or more mechanisms to prove that you are who you claim to be. Once the identity of the human or machine is validated, access is granted.

Three universally recognized authentication factors exist today: what you know (e.g. passwords), what you have (e.g. ATM card or tokens), and what you are (e.g. biometrics). Recent work has been done in trying alternative factors such as a fourth factor, e.g. somebody you know, which is based on the notion of vouching [10].

Two factor authentication is a mechanism which implements two of the above mentioned factors and is therefore considered stronger and more secure than the traditionally implemented one factor authentication system. Withdrawing money from an ATM machine utilizes two factor authentication; the user must possess the ATM card, i.e. what you have, and must know a unique personal identification number (PIN), i.e. what you know.

Passwords are known to be one of the easiest targets of hackers. Therefore, most organizations are looking for more secure methods to protect their customers and employees. Biometrics are known to be very secure and are used in special organizations, but they are not used much in secure online transactions or ATM machines given the expensive hardware that is needed to identify the subject and the maintenance costs, etc. Instead, banks and companies are using tokens as a mean of two factor authentication.

A security token is a physical device that an authorized user of computer services is given to aid in authentication. It is also referred to as an authentication token or a cryptographic token. Tokens come in two formats: hardware and software. Hardware tokens are small devices which are small and can be conveniently carried. Some of these tokens store cryptographic keys or biometric data, while others display a PIN that changes with time. At any particular time when a user wishes to log-in, i.e. authenticate, he uses the PIN displayed on the token in addition to his normal account password. Software tokens are programs that run on computers and provide a PIN that changes with time. Such programs implement a One Time Password (OTP) algorithm. OTP algorithms are critical to the security of systems employing them since unauthorized users should not be able to guess the next password in the sequence. The sequence should be random to the maximum possible extent, unpredictable, and irreversible. Factors that can be used in OTP generation include names, time, seed, etc. Several commercial two factor authentication systems exist today such as BestBuy's BesToken [15], RSA's SecurID [14], and Secure Computing's Safeword [2].

BesToken applies two-factor authentication through a smart card chip integrated USB token. It has a great deal of functionality by being able to both generate and store users' information such as passwords, certificates and keys. One application is to use it to log into laptops. In this case, the user has to enter a password while the USB token is plugged to the laptop at the time of the login. A hacker must compromise both the USB and the user account password to log into the laptop.

SecurID from RSA uses a token (which could be hardware or software) whose internal clock is synchronized with the main server. Each token has a *unique seed* which is used to generate a pseudo-random number. This seed is loaded into the server upon purchase of the token and used to identify the user. An OTP is generated using the token every 60 seconds. The same process occurs at the server side. A user uses the OTP along with a PIN which only he knows to authenticate and is validated at the server side. If the OTP and PIN match, the user is authenticated [8]. In services such as e-commerce, a great deal of time and money is put into countering possible threats and it has been pointed out that both the client and the server as well as the channel of communication between them is imperative [1].

In 2005 the National Bank of Abu Dhabi (NBAD) became the first bank in the Middle East to implement two factor authentication using tokens. It employed the RSA SecurID solution and issued its 19000 customers small hardware tokens [7, 14]. The National Bank of Dubai (NBD) made it compulsory for commercial customers to obtain tokens; as for personal customers the bank offered them the option to obtain the tokens [11]. In 2005, Bank of America also began providing two factor authentication for its 14 million customers by offering hardware tokens [5]. Many international banks also opted to provide their users with tokens for additional security, such as Bank of Queensland, the Commonwealth Bank of Australia and the Bank of Ireland [3].

Using tokens involves several steps including registration of users, token production and distribution, user and token authentication, and user and token revocation among others [6].

While tokens provide a much safer environment for users, it can be very costly for organizations. For example, a bank with a million customers will have to purchase, install, and maintain a million tokens. Furthermore, the bank has to provide continuous support for training customers on how to use the tokens. The banks have to also be ready to provide replacements if a token breaks or gets stolen. Replacing a token is a lot more expensive than replacing an ATM card or resetting a password.

From the customer's prospective, having an account with more than one bank means the need to carry and maintain several tokens which constitute a big inconvenience and can lead to tokens being lost, stolen, or broken. In many cases, the customers are charged for each token.

We propose a mobile-based software token that will save the organizations the cost of purchasing and maintaining the hardware tokens. Furthermore, will allow customers to install multiple software tokens on their mobile phones. Hence, they will only worry about their mobile phones instead of worrying about several hardware tokens.

3 Design Implementation

In this paper, we propose a mobile-based software token system that is supposed to replace existing hardware and computer-based software tokens. The proposed system is secure and consists of three parts: software installed on the client's mobile phone, server software, and a GSM modem connected to the server. The system will have two modes of operation:

1. **Connection-Less Authentication System**: A one-time password (OTP) is generated without connecting the client to the server. The mobile phone will act as a token and use certain factors unique to

it among other factors to generate a one-time password locally. The server will have all the required factors including the ones unique to each mobile phone in order to generate the same password at the server side and compare it to the password submitted by the client. The client may submit the password online or through a device such as an ATM machine. A program will be installed on the client's mobile phone to generate the OTP.

2. **SMS-Based Authentication System**: In case the first method fails to work, the password is rejected, or the client and server are out of sync, the mobile phone can request the one time password directly from the server without the need to generate the OTP locally on the mobile phone. In order for the server to verify the identity of the user, the mobile phone sends to the server, via an SMS message, information unique to the user. The server checks the SMS content and if correct, returns a randomly generated OTP to the mobile phone. The user will then have a given amount of time to use the OTP before it expires. Note that this method will require both the client and server to pay for the telecommunication charges of sending the SMS message.

A. OTP Algorithm

In order to secure the system, the generated OTP must be hard to guess, retrieve, or trace by hackers. Therefore, its very important to develop a secure OTP generating algorithm. Several factors can be used by the OTP algorithm to generate a difficult-to-guess password. Users seem to be willing to use simple factors such as their mobile number and a PIN for services such as authorizing mobile micropayments [9, 14]. Note that these factors must exist on both the mobile phone and server in order for both sides to generate the same password. In the proposed design, the following factors were chosen:

1. **IMEI number:** The term stands for International Mobile Equipment Identity which is unique to each mobile phone allowing each user to be identified by his device. This is accessible on the mobile phone and will be stored in the server's database for each client.

2. **IMSI number:** The term stands for International Mobile Subscriber Identity which is a unique number associated with all GSM and Universal Mobile Telecommunications System (UMTS) network mobile phone

users. It is stored in the Subscriber Identity Module (SIM) card in the mobile phone. An IMSI is usually 15 digits long. The first 3 digits are the Mobile Country Code, and are followed by the Mobile Network Code (MNC). The remaining digits are the mobile subscriber identification number (MSIN) within the network's customer base. This number will also be stored in the server's database for each client.

3. **Username:** Although no longer required because the IMEI will uniquely identify the user anyway. This is used together with the PIN to protect the user in case the mobile phone is stolen.
4. **PIN:** This is required to verify that no one other than the user is using the phone to generate the user's OTP. The PIN together with the username is data that only the user knows so even if the mobile phone is stolen the OTP cannot be generated correctly without knowing the user's PIN. Note that the username and the PIN are never stored in the mobile's memory. They are just used to generate the OTP and discarded immediately after that. In order for the PIN to be hard to guess or brute-forced by the hacker, a minimum of 8-characters long PIN is requested with a mixture of upper- and lower-case characters, digits, and symbols.
5. **Hour:** This allows the OTP generated each hour to be unique.
6. **Minute:** This would make the OTP generated each minute to be unique; hence the OTP would be valid for one minute only and might be inconvenient to the user. An alternative solution is to only use the first digit of the minute which will make the password valid for ten minutes and will be more convenient for the users, since some users need more than a minute to read and enter the OTP. Note that the software can modified to allow the administrators to select their preferred OTP validity interval.
7. **Day:** Makes the OTP set unique to each day of the week.
8. **Year/Month/Date:** Using the last two digits of the year and the date and month makes the OTP unique for that particular date.

The time is retrieved by the client and server from the telecommunication company. This will ensure the correct time synchronization between both sides.

The above factors are concatenated and the result is hashed using SHA-256 which returns a 256 bit message. The message is then XOR-ed with the PIN replicated to 256 characters. The result is then Base64 encoded which yields a 28 character message. The message is then shrunk to an administrator-specified length by breaking it into two halves and XOR-ing the two halves repeatedly. This process results in a password that is unique for a ten minute interval for a specific user. Keeping the password at 28 characters is more secure but more difficult to use by the client, since the user must enter all 28 characters to the online webpage or ATM machine. The shorter the OTP message the easier it is for the user, but also the easier it is to be hacked. The proposed system gives the administrator the advantage of selecting the password's length based on his preference and security needs.

A detailed breakdown of the generation process is illustrated through the example below. Assume the following system parameters:

1. Username = N81z
2. PIN = 3690
3. IMEI = 123456789123456
4. IMSI = 12345678912345
5. Time of Generation = 7/6/2009 11:02 AM

1. The factors are concatenated giving the following:
   110Sun0967N81z123456. . . ..7891234563690e6a(length depends on the length of the username and PIN)
2. The result is hashed using SHA-256 to give:
   E6D15D80FB90A1801235. . . . . . E8F23DBB815E12
3. The PIN is also hashed using SHA-256 to give:
   6A0E00006A0E00006A0E00006. . . . . . 00006A0E0000
4. The two hashes obtained above are XOR-ed to give:
   8CDF5D80919EA180783BCA2. . . . .DD18F5E12
5. We then shrink the above hash to half its length by XOR-ing the left-most byte with the rightmost byte and so on to obtain:
   9E81D251AC6C47A50B4D2EE38799B666

6. The result is then XOR-ed with the PIN hash again to give: F48FD251C66247A561432EE3ED97B666

7. The result is then shrunk again using the same byte wise XOR-ing process and then rehashed with the PIN. This process is repeated till the hash is shrunk to the desired length which in our case was 8 characters and gives: 3C330999

8. The result is then encoded using base-64 encoding to give the final OTP:PDMJmQ==

B. Client Design

Figures 1 and 2 describe the connection-less and the SMS-based methods for generating the OTP.

A J2ME program is developed and installed on the mobile phone to generate the OTP. The program has an easy-to-use GUI that is developed using the NetBeans drag and drop interface. The OTP program has the option of generating the OTP locally using the mobile credentials, e.g. IMEI and IMSI numbers, or requesting the OTP from the server via an SMS message. The default option is the first method which is cheaper since no SMS messages are exchanged between the client and the server. However, the user has the option to select the SMS-based method.

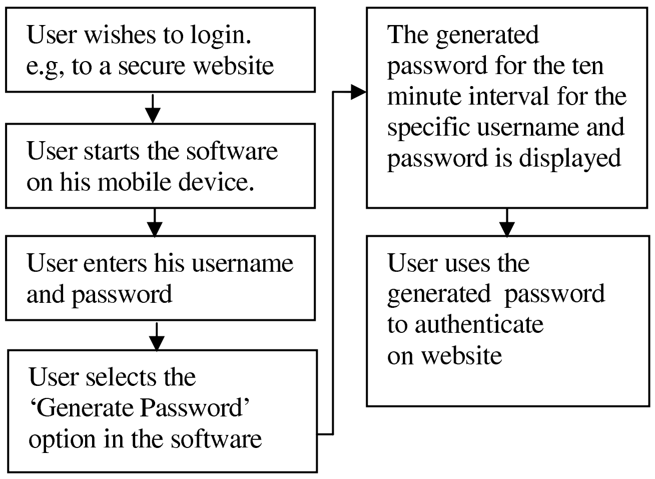

User wishes to login. e.g, to a secure website
User starts the software on his mobile device.
User enters his username and password
User selects the 'Generate Password' option in the software
The generated password for the ten minute interval for the specific username and password is displayed
User uses the generated password to authenticate on website

Figure 1: The connection-less process of generating the OTP.

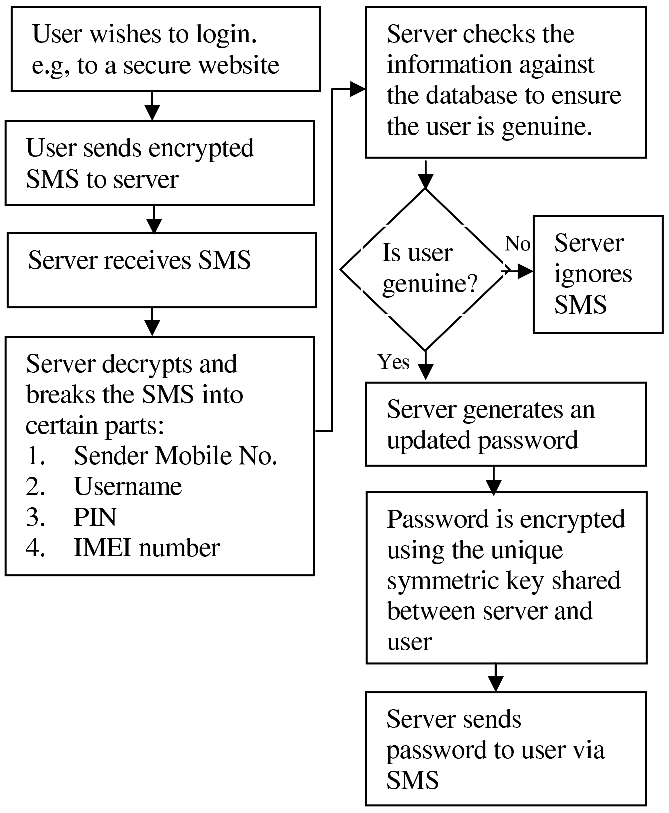

User wishes to login. e.g, to a secure website
User sends encrypted SMS to server
Server receives SMS
Server decrypts and breaks the SMS into certain parts:
1. Sender Mobile No.
2. Username
3. PIN
4. IMEI number
Server checks the information against the database to ensure the user is genuine.
Is user genuine?
No
Server ignores SMS
Yes
Server generates an updated password
Password is encrypted using the unique symmetric key shared between server and user
Server sends password to user via SMS

Figure 2: The SMS-based process of generating the OTP.

In order for the user to run the OTP program, the user must enter his username and PIN and select the OTP generation method. The username, PIN, and generated OTP are never stored on the mobile phone.

If the user selects the connection-less method the username and PIN are used to locally generate the OTP and are discarded thereafter. The username and PIN are stored on the server's side to generate the same OTP.

If the user selects the SMS-based method, the username and PIN, in addition to the mobile identification factors, are encrypted via a 256-bit symmetric key in the OTP algorithm and sent via an SMS to the server. The server decrypts the message via the same 256-bit symmetric key, extracts the identification factors, compares the factors to the ones stored in the database, generates an OTP and sends the OTP to the client's mobile phone if the factors are valid. The advantage of encrypting the SMS message is to prohibit sniffing or man-in-the-middle attacks. The 256-bit key will be extremely hard to brute-force by the hacker. Note that each user will have a pre-defined unique 256-bit symmetric key that is stored on both the server and client at registration time.

Figure 3 shows a sample screenshot of the client software and the generated OTP password. Since the program is written in J2ME it can be installed on any J2ME-enabled mobile device.

C. Database Design

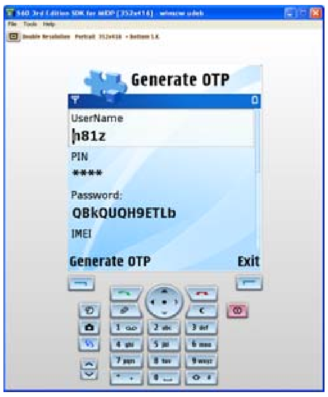
A database is needed on the server side to store the client's identification information such as the first name, last name, username, pin, password, mobile IMEI number, IMSI number, unique symmetric key, and the mobile telephone number for each user. The password field will store the hash of the 10 minute password. It will not store the password itself. Should the database be compromised the hashes cannot be reversed in order to get the passwords used to generate those hashes. Hence, the OTP algorithm will not be traced.

D. Server Design

A server is implemented to generate the OTP on the organization's side. The server consists of a database as described in Section 3.C and is connected to a GSM modem for SMS messages exchange.

The server application is multithreaded. The first thread is responsible for initializing the database and SMS modem, and listening on the modem for client requests. The second thread is responsible for verifying the SMS information, and generating and sending the OTP. A third thread is used to compare the OTP to the one retrieved using the connection-less method.

In order to setup the database, the client must register in person at the organization. The client's mobile phone identification factors, e.g. IMEI, and SIM card info, e.g. IMSI, are retrieved and stored in the database, in addition to the username and PIN. The J2ME OTP generating software is installed on the mobile phone. The software is configured to connect to the server's GSM modem in case the SMS option is used. A unique symmetric key is also generated and installed on both the mobile phone and server. Both parties are ready to generate the OTP at that point.



UserName
h81z
PIN
****
Password:
QBkQUQH9ETLb
IMEI

Figure 3: Snapshot of the client OTP program.

4 System Testing

The server was implemented using Java. A Siemens MC35i GSM modem was used for sending and receiving SMS messages on the server side. The smslib3.2.0 [17] library was used to send the messages and the SHA 4j [16] library was used to hash the password. An Oracle 10g was used as a database.

The client was implemented using J2ME and installed on a Nokia E60 and Nokia E61 phone. Both methods, the connection-less and SMS-based, were tested. In the first method, fake user accounts were setup on both the mobile phone and server. The mobile phone was used to generate 5000 random OTPs at various times of the day and all 5000 generated OTPs matched the OTPs generated on the server side. The use of date and time from the telecommunication company helped solve the synchronization problem. Furthermore, using the first digit of the minute gave the user enough time, i.e. 10 minutes, to compute, read, enter, and send the OTP.

The second SMS-based method was also tested. Once the client requests an OTP via an SMS, the server would check the u ser credentials, generate the OTP, and send it back instantly. The SMS message was also encrypted using a 256-bit unique symmetric key to avoid man-in-the-middle attacks. The complete process of receiving the request, checking the factors, generating and sending the OTP was done in less than a second from the server's side. This confirms the efficiency and effectiveness of the proposed system when dealing with many customers.

An experiment was done to check the chances of generating two identical hashes for two different users, i.e. generating a hash collision. In this experiment, the database was filled with information for ten fake users. The generated OTP was set to be 14 characters long. The OTP can consist of upper-case and lower-case characters, digits, and symbols, yielding a total of 4.9E+91 different possible combinations. This is almost impossible to brute force with the existing computing facilities. For each of the ten users, a 100,000 OTPs were generated (1 OTP was generated every second). A total of 1 million OTPs were generated and compared. All million OTPs were unique.

5 Conclusions

Today, single factor authentication, e.g. passwords, is no longer considered secure in the internet and banking world. Easy-to-guess passwords, such as names, age, and date of birth, are easily discovered by automated password-collecting programs. Two factor authentication has recently been introduced to meet the demand of organizations for providing stronger authentication options to its users. In most cases, a hardware token is given to each user for each account. The increasing number of carried tokens and the cost the manufacturing and maintaining them is becoming a burden on both the client and organization. Since many clients carry a mobile phone today at all times, an alternative is to install all the software tokens on the mobile phone. This will help reduce the manufacturing costs and the number of devices carried by the client.

This paper focuses on the implementation of two-factor authentication methods using mobile phones. It provides the reader with an overview of the various parts of the system and the capabilities of the system. The proposed system has two option of running, either using a free and fast *connection-*

less method or a slightly more expensive *SMS-based* method. Both methods have been successfully implemented and tested, and shown to be robust and secure. The system has several factors that makes it difficult to hack, such as:

1. At least 10 factors related to the mobile phone, SIM card, user, date, and time are used to generate a difficult-to-guess, unique one-time password (OTP).
2. The system can easily run on any J2ME-enabled phone.
3. The system has a user-friendly GUI.
4. The OTP is only generated on the registered mobile phone and SIM card for stronger authentication.
5. Even if the mobile phone is stolen an 8-characters PIN must be input to the phone to generate the correct OTP which is hard to brute force or guess by the hacker.
6. The PIN or OTP are never stored on the client's mobile phone for additional security.
7. The OTP can be valid for an administrator-specified time interval based on the security needs.
8. The OTP length is also set by the administrator based on the security needs.
9. The OTP generation can be done for *free* on the client's side without the need to connect to the server. An alternative cheap option is also introduced using SMS.
10. In case the SMS option is used, all exchanged SMS messages are encrypted using a 256-bit key to avoid man-in-the-middle attacks.
11. The time/date synchronization problem between the client and server is solved by retrieving the time and date from the telecommunication company.

Future developments include a more user friendly GUI and extending the algorithm to work on Blackberry, Palm, and Windows-based mobile phones. In addition to the use of Bluetooth and WLAN features on mobile phones for better security and cheaper OTP generation.

References

[1] A. Jsang, G. Sanderud, Security in Mobile Communications: Challenges and Opportunities, in *Proc. of the Australasian information security workshop conference on ACSW frontiers*, 2003, 43-48.

[2] Aladdin Secure SafeWord 2008. Available at http://www.securecomputing.com/index.cfm?skey=1713

[3] A. Medrano, Online Banking Security-Layers of Protection. Available at http://ezinearticles.com/?Online-Banking-Security-Layers-of-Protection&id=1353184

[4] B. Schneier, Two-Factor Authentication: Too Little, Too Late,in *Inside Risks 178*, *Communications of the ACM*, **48**, April 2005.

[5] D. Ilett, US bank gives two-factor authentication to millions of customers, 2005. Available at http://www.silicon.com/financialservices/0,3800010322,39153981,00.htm

[6] D. de Borde, Two-factor authentication,*Siemens Enterprise Communications UK-Security Solutions*, 2008. Available at http://www.insight.co.uk/files/whitepapers/Twofactor/authentication/(White/paper).pdf

[7] D. Milne, NBAD's internet bankers get RSA, 2005. Available at http://www.itp.net/index.php?view=article&id=484322&Itemid=1 &option=com_content

[8] D. Shinder, Why and how to implement SecurID Authentication. Available at http://www.windowsecurity.com/articles/Why-how-implement-SecurID-uthentication.html

[9] A. Herzberg, Payments and banking with mobile personal devices, *Communications of the ACM,* **46**, May 2003, 53-58.

[10] J. Brainard, A Juels, R. L. Rivest, M. Szydlo, M. Yung, Fourth-Factor Authentication: Somebody You Know, *ACM CCS*, 2006, 168-78.

[11] NBD Online Token. Available at http://www.nbd.com/NBD/NBD_CDA/CDA_Web_pages/ Internet_Banking/nbdonline_topbanner

[12] N. Mallat, M. Rossi, V. Tuunainen, Mobile banking services, *Communications of the ACM,* **47**, May 2004, 42-46.

[13] RSA SecurID Authentication: A Better Value for a Better ROI, 2001. Available at http://www.opsec.com/solutions/partners/downloads/rsa_securid_roi.pdf

[14] RSA Security Selected by National Bank of Abu Dhabi to Protect Online Banking Customers. Available at http://www.rsa.com/press_release.aspx?id =6092

[15] Ryan Groom, Two Factor Authentication Using BESTOKEN Pro USB Token. Available at http://bizsecurity.about.com/od/mobilesecurity/a/twofactor.htm

[16] Sha4J. Available at http://www.softabar.com/home/content/view/46/68/

[17] SMSLib. Available at http://smslib.org/