A Secure Mobile-based Authentication System for e-Banking

Helena Rifà-Pous

Department of Computer Sciences, Multimedia and Telecommunication,
Universitat Oberta de Catalunya (UOC), 08018-Barcelona, Spain
hrifa@uoc.edu

Abstract. Financial information is extremely sensitive. Hence, electronic banking must provide a robust system to authenticate its customers and let them access their data remotely. On the other hand, such system must be usable, affordable, and portable. We propose a challenge-response based one-time password (OTP) scheme that uses symmetric cryptography in combination with a hardware security module. The proposed protocol safeguards passwords from keyloggers and phishing attacks. Besides, this solution provides convenient mobility for users who want to bank online anytime and anywhere, not just from their own trusted computers.

Keywords: one-time password, challenge-response authentication, mobile security, attacks.

1 Introduction

Financial institutions that offer Internet banking services must have effective and reliable methods to authenticate customers. Password authentication is the most spread authentication mechanism over generic web sites on the Internet. Its success is because it is simple, easy to implement in any application, and portable. However, it is not a sufficiently secure solution to be used in financial services.

Part of the security issues of password-based solutions are due to the own nature of the scheme, which is very simple and so, vulnerable to brute force attacks. However, some other problems appear from employing bad practices. In [1], Florencio and Herley have done a large scale study of authentication users practices and concluded the average consumer prioritizes simplicity in front of security.

As the number of services a user is registered grows, so does the number of password he needs to manage. Facing the problem to remember so many passwords, users tend to maintain a small collection of passwords and reuse them in several sites. Besides, the strength of the passwords is, in the majority of the cases, low. Password management systems can make user's work easier.

However, they also represent an excellent point of attack since if their security protection is broken, all user's accounts are compromised.

This paper addresses the problem of having a web authentication scheme that is secure as well as easy to use, employing the mobile phone as an authentication token. We propose a One-Time Password (OTP) system. Contrary to other OTP schemes that use specialized hardware, our model can be deployed to any mobile that is Java enabled.

The protocol provides the following benefits:

- A secure password repository. Users do no longer need to remember a password for each account they have, and so, they accounts are inherently more secure since the login key is not shared between multiple sites. Moreover, the repository is protected from thefts.
- Keylogging protection on the PC. A client PC does not have access to the secret service key a user shares with a web site, so malware, spyware [2] or even keyloggers [3] can not capture it.
- A portable solution. We propose a web authentication protocol that is based on mobiles, but the data to transmit to/from the computer is so small than user can do it by hand. No personal area network needs to be set up, and so, the protocol can be securely used from any public place (Internet café, library, etc.)

This paper is organized as follows. First we give a brief overview of basic Internet authentication models. Next we review other systems that use the mobile as a hardware token to compute OTPs. In section 4 we describe the proposed scheme and detail the protocol. Section 5 evaluates the security of our model. Finally, we conclude the paper in section 6.

2 Internet authentication schemes

Online authentication techniques are based on one or a combination of the following authentication proofs [4]:

- Something the user knows (e.g., password, PIN)
- Something the user has (e.g., ATM card, smart card)
- Something the user is (e.g., biometric characteristic, such as a fingerprint)

The level of security an authentication methodology provides varies upon both the employed authentication proofs and the manner in which the protocol is deployed. In general, they are stronger as more different authentication proofs are required.

In the U.S., federal regulating agencies[1] consider single-factor authentication, that is, protocols that only involves one authentication proof, as inadequate

[1] Board of Governors of the Federal Reserve System, Federal Deposit Insurance Corporation, National Credit Union Administration, Office of the Comptroller of the Currency, and Office of Thrift Supervision

for the risks and services associated with Internet banking [5]. Yest, one-factor authentication schemes are still broadly used in home banking.

The simplest and most used one-factor authentication scheme is the username and password. The communication channel between the consumer and the server is usually protected from eavesdropping using the SSL protocol. In this way, data transmitted remains confidential. However, in spite of SSL, password authentication schemes are vulnerable. An attacker can easily capture a user's password by installing a keylogger program on a client PC [3], or getting it on a phishing website [6]. The method is also susceptible to dictionary attacks, which attempt to guess passwords based on words in the dictionary.

To improve the security of password-based authentication systems, one strategy it has been to require the use of strong passwords, that is, long text strings that are hard to guess because they include alphanumeric characters as well as punctuation symbols. However, the difficulties to remember tedious passwords make the system even weaker. Final users are frequently clients of several services and for each of them they have an independent account. Managing the information of all of them is complex so users tend to put the same password to various services or they write them down in an insecure place [1]. These lead to vulnerabilities that can compromise the system. Nevertheless, this method is not robust against replay or phishing attacks.

Other more secure software solutions are the ones based on software PKI client certificates. These systems are practical in terms a user has only to remember one thing, the password of his key store. However they have the drawback of security and portability. If key stores are installed on the PC, they are vulnerable to off-line credential-stealing attacks. On the other hand, users can only access the services using the computer in which they have their keys installed. Users that do not have a computer and connect to the web from an Internet café, can not make use of client certificates.

Nowadays, the most used two factor authentication scheme for Internet banking is scratch cards. A bank scratch card is like a lottery scratch card, carrying a grid of numbers needed to access an account. When a user wants to access his bank Internet account, he is asked for his password (something he knows) and the characters contained in a randomly chosen cell in the grid of this card (something he has). This system is stronger than the simple username and password authentication, but it is not strong enough for financial services. The scratch cards have a very limited number of cells and thus, it is easy for an eavesdropper to replicate the card after listening to several Internet banking sessions.

Another two factor authentication system is based on PKI certificates on smart cards. The drawback of this scheme is that the client PC needs to have a smart card reader. This is not usual in the majority of computers on Internet cafés, hotels, libraries, or any other places of public access. Hence, this model lacks of usability.

A more flexible approach is using a PKI based-system on a hardware-based encryption module. The implementation on Digital Signal Processors (DSP)

leads to a fast and secure solution [7, 8]. However, the problem is that the user needs to acquire a specific appliance and carry it with him always.

Besides, PKI schemes also faces another problem. If companies do not accept certificates issued by some central organization, the management of the system becomes tedious. Users have a smart card or a hardware token for storing the keys of each particular web site they deal with, and so, they have to deal with a lot of devices.

One Time Password (OTP) authentication is a method to reduce the potential for compromised user credentials using login passwords that are only valid once. Even if an attacker is capable of sniffing the password that a user has employed to enter in a site, it is of no use since the password is no longer valid. Moreover, it is extremely difficult to predict the next password based on the previous one.

A password-generating token produces a unique OTP each time it is used. The function that generates such passwords must be non invertible. There are three types of schemes to generate one-time passwords:

- Based on time, such as SecurId [9]. Time-synchronization is required between the authentication server and the client providing the password
- Based on a challenge (e.g. a random number chosen by the authentication server or transaction details) and a counter
- Based on some internal data (e.g. the previous password) or counter (e.g. systems based on hash chains, such as S-Key [10])

We focus our work on OTP generating systems that are based on a challenge.

3 Related Work

There are some OTP solutions based on a mobile phone. In [11, 12] a multi-channel communication is used (Internet and GSM) in order to improve the security of the authentication scheme. In [11] a user logs in the web site using a username and a password. Then, a one time password is sent via SMS to his mobile, and the user enters this data in the web authentication form. If it is correctly verified, the user is authenticated into the application. In this system the mobile is used as a mere point of reception, not as a hardware token that stores and computes keys.

On the other hand, in [12] what is sent though the GSM channel is a challenge. The mobile computes a one-time password using this challenge and sends it to his computer through a bluetooth connection. Finally, the password is forwarded to the server.

The main trouble of these two schemes that rely on SMS messages to perform the authentication is that the session establishment between the user and the server is slow because SMS messages are not real-time. Thus, the system is not practical. On the other hand, users may want to connect to their Internet bank accounts from places in which there is no cellular connectivity (in some sensitive environments GSM signals are blocked), and these models do not allow it.

Other OTP solutions [13, 14] deal with a password generation in the mobile using as input a server challenge sent through the Internet connection. Once in the PC, the challenge is transfered to the mobile using a bluetooth channel. The problem is that bluetooth is usually not available from public access computers. Besides, it presents some relevant vulnerabilities and threats [15, 16] -most of which due to faulty implementations- that jeopardize the system.

Some other OTP mobile schemes focus on the speed of the process and base the generation of the one-time password on a time factor (no server challenge is needed). This is the case of the Free Auth Project [17]. The inconvenient of using this approach in a mobile context is the required time synchronization between the mobile and the server. Users roughly configure the clock of the mobile phone when they travel, and they are not very much concerned on setting the correct time zone. Hence, protocols based on absolute time are not feasible.

The MP-Auth scheme [18] uses the mobile as a secure device to store keys and encrypt passwords for web authentication. It is a one factor authentication mechanism that safeguards passwords from keyloggers, phishing attacks and pharming. Nevertheless, if an attacker learns a user password he can impersonate that user.

4 Mobile OTP Scheme

We present an OTP scheme that comprises a web server, a browser, and a client application on a cellphone. The protocol uses the mobile as a security hardware to store the secret keys that allow getting the OTPs. Data transmission between the mobile and client PC is simple, so it does not need to be hold by a communication channel like bluetooth; it can be entered using keypads. Figure 1 overviews the architecture.

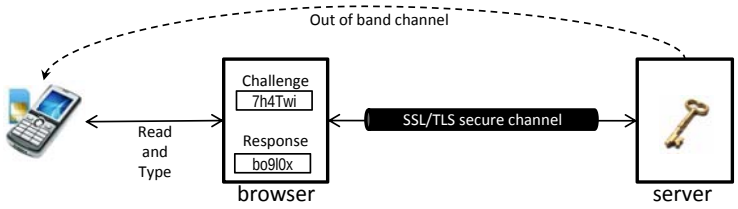

Fig. 1. System Architecture

Our OTP scheme consists of three phases, namely, registration phase, authentication phase and transaction approval phase. In the registration phase, the server issues initialization keys to the users who request registration and meet the requirements. They store these keys in their mobiles. After a successful registry, the user can access the web services through the authentication phase. In the authentication phase, he sends his identifier to the server, which replies

U	The user of the computer system that uses the authentication protocol to login to the host
M	The mobile phone
S	The authentication server
UID	User Identifier
SID	Service Identifier
PIN	Personal Identification Number
ch	Challenge
k_t	Temporal service key
k_s	Service key
k_o	Old service key
$H(m)$	Public one-way hash function of message m, i.e. SHA-1
$T(x, m)$	Truncation function at x bits on message m
OTP	One Time Password
SAC	Server Authentication Code
TA	Transaction Authorization

Table 1. Notation

with a challenge. Then, the user, using the mobile phone, computes a new one-time password and sends it to the server. The server verifies the validity of the submitted token and determines whether the user is accepted or not. If it is, the server sends a server authentication code to the user in order he can verify its authenticity. Finally, the user compares the code received from the server with the one computed by his mobile to verify the identity of the server whom he is connected.

The initial key of the system is short (30 bits of effective length). However, since sharing a short static session key with the server is not recommended for security issues, this key is updated regularly, in each login. Besides, the server can occasionally mandate a key renewal to some users that rarely login in the system or do it always from the same computer.

The fact that the key is renewed periodically has the drawback of synchronization. Several facts can jeopardize the events synchronization between the client and the server: a network failure, the client does not introduces the OTP to the web, etc. So that, the server always maintains the last verified service key in a record. If a client logins after a break-down session using this old service key, he is accepted in the site. If a client logins three consecutive times with the same service key, the bank blocks his online service account.

The authentication phase is performed every time the user wants to login and gives reading access to the site. On the other hand, when he wants to carry out a sensible transaction (i.e. a transfer to another user's account), he is asked for a validation token to approve the operation. This reinforces the system and prevents any impersonation attack (see details on security analysis in section 5).

Following, we describe these three phases in turn. The notation used in the protocols is described in Table 4.

4.1 Registration Protocol

We assume users that register to the system posses a mobile phone. In this phase, the mobile is set up to work as a web server's authentication hardware device.

Before executing the registration protocol, the server must verify the identity and attributes of the requester user. This process is usually performed face to face, although depending on the service and context, other options may be valid. For example, the remotely authentication and identification through a national ID card. If the user is admitted, the registration protocol takes place. The details are described in the following steps.

1. $[S \rightarrow U]$ S sends to U the following initialization data through an out of band channel (face tot face, sms, postal mail, phone, etc.)
   UID: User identifier in the system. It is an easy to memorize identifier that can be chosen either by the server or the final user.
   SID: Server identifier. A brand name or short text string that identifies the server.
   k_0: Initial service key. It is coded in base32 and it is eight characters long. The first six characters (30 bits) a randomly generated. The last two ones are a checksum.
2. S creates and maintains a registry with the data of subscribed clients, in the following way:
   $$User := UID, \quad k_s := H(k_0)$$
3. U opens the OTP application on his mobile phone. He is asked for a PIN to get access to the application. Then, he creates a new server account with the data received from the server. The mobile creates an entry in its table with the data:
   $$Server := SID, \quad User := UID, \quad k_s := H(k_0)$$
   The checksum embedded in the service key k_0 prevents the user to store a wrong key in the mobile.

4.2 Authentication Protocol

The mobile must produce the appropriate OTP from the stored service key and the challenge provided by the server. Then the server verifies the received OTP, replies with an authentication code SAC to the client, and updates the service key with the information of the last challenge. Finally, the client checks the authenticity of the server using the SAC, and if correct, updates the service key. Figure 2 shows the protocol steps.

OTPs are generated computing a XOR of the service key and the hash of the challenge. The result of this operation is a temporal service key which is truncated to 128 bits to be used as an AES-128 key. Then, the user identifier UID is ciphered with an AES cryptosystem using this key. The result of the encryption truncated to 42 bits is the OTP.

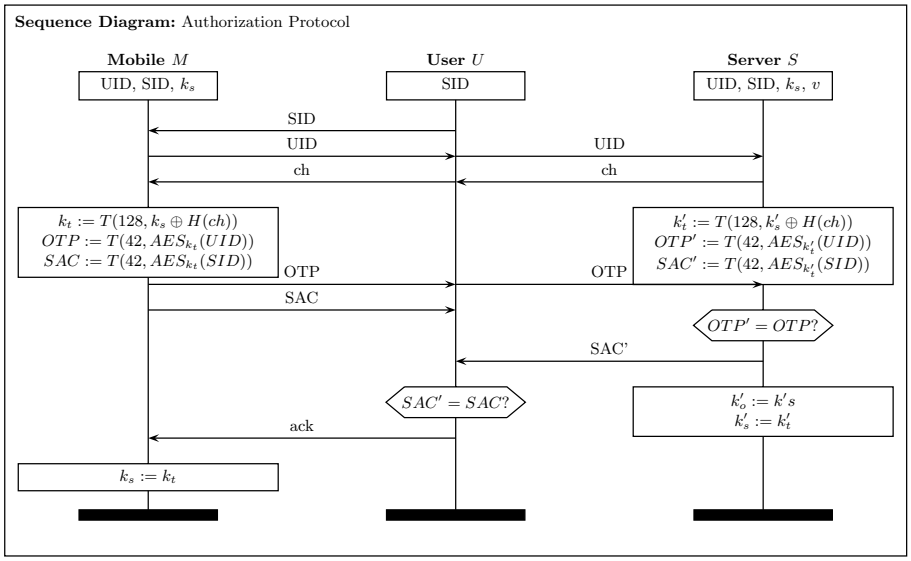

Fig. 2. Diagram of Authentication Protocol

1. $[U \leftrightarrow S]$ U opens his browser and connects to the web banking site of server S via SSL/TLS with server side authentication. U checks the server certificate is valid.
2. U opens the OTP application on his mobile phone. He is asked for a PIN to get access to the application. Then, the mobile displays a list of server identifiers registered in the mobile. U choses the SID of his Internet banking server S.
3. The mobile displays the UID of the user U that identifies him in the server S and waits for a challenge.
4. $[U \rightarrow S]$ U enters his UID in the bank login web form and sends it.
5. $[S \rightarrow U]$ S replies with a challenge ch that is eight characters long. ch is coded in base32 to facilitate its management using the mobile phone keypad. The first six characters (30 bits) a randomly generated. The last two ones are a checksum.
6. U introduces ch in the mobile application.
7. The mobile computes the OTP and SAC. To compute these values it uses the stored service key k_s, the service identifier SID, and the provided challenge ch.
   First it computes a temporal service key k_t

$$k_t := T(128, k_s \oplus H(ch)) \tag{1}$$

   The mobiles encrypts the user UID with an AES-128 cipher. The output is truncated to 42 bits and encoded in ascii-7 in order to be an appropriately

presentable response string. This is the OTP, which is 6 characters long.

$$OTP := T(42, AES_{k_t}(UID)) \tag{2}$$

The SAC is computed in a similar way than the OTP, and is used to verify the authenticity of the server. It prevents the appearance of fake servers that desynchronize the parallel generation of service keys in the consumer and the server.

$$SAC := T(42, AES_{k_t}(SID)) \tag{3}$$

The mobile displays the OTP and the SAC in its screen. U manually copies the OTP into the bank's web form.

8. S calculates the OTP using equations 1 and 2, and checks if the received OTP is correct. Because the generation of the OTP depends on the previous shared service key and this key can diverge due to unsuccessful logins, the server tries to synchronize its local temporal key k_t with the one in the client using the last valid service key k_o.
9. S updates the service key k_s and the old service key k_o

$$k_o := k_s \tag{4}$$

$$k_s := k_t \tag{5}$$

10. $[S \rightarrow U]$ S computes its authentication code SAC using equation 3 and sends it to the client.
11. U checks if the received SAC is the same as the mobile displays. If it is, U confirms the server authentication in the mobile, and the mobile updates the service key as in equation 5.

Once an authenticated channel is established between the user and the bank, the user can consult the data of his account and perform all kind of transactions. In order to thwart session attacks, the bank can require the user submits an approval of the transaction order.

4.3 Transaction Approval

When a user wants to launch a financial transaction, the following steps take place.

1. $[S \rightarrow U]$ S sends to U a TID, which is a 4 numeric digits random number that identifies the transaction.
2. $[U \rightarrow S]$ U enters the TID in the mobile using the keypad. The mobile computes:

$$TA := T(20, AES_{k_t}(TID)) \tag{6}$$

TA is encoded in base32, so it is 4 characters long. U copies TA value in the web form and sends it to S.

3. S checks the authenticity of TA comparing the received value with the one it computes using equation 6. If they are equal, S confirms the user with which it is connected is the legitimate one.

4.4 Implementation Issues

We developed a prototype to evaluate the usability of the proposal. The service has been set in an Apache Web Server using Java Servlets. The client application is a MIDlet that has been programmed for the CLDC configuration of Sun Java Wireless Toolkit (formerly known as Java 2 Platform Micro Edition (J2ME) Wireless Toolkit) in order to be suitable for mobile phones.

In order to create a secure key store in the mobile, there are two alternatives:

1. Ciphered data base: The secret key that the user shares with the server site and that is the seed to compute the one-time password, is ciphered and stored in the mobile.
2. SWIM card: SWIM are SIM cards that contain a Wireless Identity Module (WIM). Such module is tamper resistant, which means that provides a very strong guarantee that certain sensible information can not be exported outside the hardware. The SWIM card is capable of storing keys and performing cryptography. It is required to enter the PIN code of the mobile to gain access to use the keys stored in the WIM.

Using SWIM cards provides efficiency and security, and it is recommended for e-banking applications. However, the majority of the consumers do not have SWIM cards nowadays. Besides, to take profit of the features of the SWIM card from a MIDlet application, it is also required to have a mobile phone that supports the Java Specification Request (JSR) 177 [19]. JSR 177 defines the Security and Trust Services API (SATSA) for the Sun Java Wireless Toolkit. In particular, the PKI package of the SATSA library supports WIM and provides methods to manage the user's credentials and cipher and sign messages. Few mobile phones in use today have these characteristics.

On the other hand, software secure stores in the SIM are slightly slower and not so robust. In general, the concern for security in software stores is because the access is protected from unauthorized users using a PIN or password, which is vulnerable to brute force or guessing attacks. However, the implementations of a secure store in the mobile are not so vulnerable as in a PC because the mobile is a personal device that usually travels with its owner. So, an attacker needs to stole the phone to reach the store and break its security.

In order to develop an application that can be used by any bank client that has a mobile phone, it may be desirable to offer the two implementation alternatives to the final users. Here we present an implementation that uses the second approach, a ciphered data base, since it is the one that can present some challenges.

The requirements for the mobile device are just that it is Java enabled. We used the OpenBaseMovil [20] library to implement the secure store in the MIDlet. For cryptographic operations we used the Bouncy Castle Lightweight Crypto API [21].

The user has to create a PIN for the store the first time he uses it. Then, every time it starts the application, it is challenged for the PIN. The application controller stores a hash of the PIN so that the user input is checked against it. If

the PIN is correct, it is used to decrypt an internal key, which in turn is used to decrypt the sensible data of the secure store. The fact of using a key to encrypt and decrypt the store and keep it encrypted with a user's PIN, is to facilitate that users can change their PIN when they like. When the PIN is changed, only the encrypted version of the internal key has to be updated, but not all the entries of the secure store.

If a user loses or gets the mobile stolen, he has to block his Internet banking account access. Financial institutions must facilitate this procedure through secure channels, in a similar way in which a registry process in the Internet Bank takes place.

5 Security Analysis

In this section, we discuss the security properties of our scheme.

The users credentials are stored in the mobile and can only be accessed by someone that has physical access to the device. If the key store is located in the SWIM card, which is recommended because the SWIM is a tamper-resistant device, malicious software can not get the user's sensible data or related functionality. If the key store is software-based, the mobile owner has to be cautious in not exposing the terminal neither to loses nor temporal thefts. If an attacker has physical access to the mobile and can make a copy of its whole memory, he will be able to perform off-line attacks (the most dangerous in this case is the dictionary attack) and at last, break the security of the key store and obtain the service key.

Nevertheless, notice that the service key that the user and the server share, it is updated every time the user accesses his Internet bank account. The new internal key depends on the challenge that the server sends to the consumer, and so, an attacker that has been able to decrypt a user's key store, can not take profit of this information if internal keys have already been updated. Thus, if a user considers he has left his mobile unattended and can be victim of an attack, he has to connect to his Internet bank account. If he can enter normally, there is no security threat. If not, he has to cancel his online account and get a new service key.

For synchronization issues, the protocol allows the use of the same service key in three consecutive login attempts. This does not threaten the system since it is only applied when a session exchange has not been completed (the server has not received any client package after sending him its authentication code SAC). Hence, when a user successfully enters in his online bank account, the service key is updated and he can be sure that nobody can impersonate him in the web.

On the other hand, login passwords to access the Internet banking accounts are used only once. Challenges are generated each time a user requests a login, and are only valid during a limited period of time. Hence, although an attacker can sniff the user-server communication, he cannot reuse the information for a later login. The system is protected against replay and phishing attacks.

If we are working on an untrusted PC that has malware installed, a malicious application could capture all keyboard input and send it to a predefined address even before this information goes to the real destination server. In this case, the intruder could use this information to get a punctual access to the user's bank account. However, when the legitimate user would finally send the login information to the server, the server would detect a duplicated response and would challenge the user again. Servers only maintain a working session with the client, so if the legitimate user is challenged again an successfully authenticated, the other fake working session is blocked.

In any case, although an intruder can get access to a bank using this exploit, he is not able to perform any transaction. All financial transactions require explicit user acceptance, and the malicious user does not possess the service key to compute such authorization.

Finally, the scheme is also protected to man-in-the-middle attacks since we use a server authenticated SSL transmission channel between the consumer and the web server.

6 Concluding Remarks

Financial institutions offering Internet services must have reliable and secure methods to authenticate their customers. The agencies advocate two-factor authentication mechanisms to guarantee the identity of the accessing users.

We have proposed a practical, effective, and secure two-factor authentication scheme based on OTP. The system relies on something the user knows (the PIN to access the OTP data on the mobile) and something the user has (the mobile). We use the phone as a hardware token since it is a personal device that consumers already have on them an with which are confident. Thus, the deployment of the system is affordable to the general public and minimizes risk for all parties.

The OTP application is usable from any Java mobile. It does not need bluetooth connectivity, neither in the registry phase nor in the authentication. Hence, it can be used in any public terminal. On the other hand, contrary to hardware token systems, it securely gathers the credentials for multiple web servers and so, it is a portable solution.

Acknowledgements

This work is partially supported by the Spanish Ministry of Science and Innovation and the FEDER funds under the grants TSI-020100-2009-374 SAT2, TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER CSD2007-00004 ARES.

References

1. Florencio, D., Herley, C.: A large-scale study of web password habits. In: Proc. of the International Conference on World Wide Web (WWW), New York, NY, USA, ACM (2007) 657–666

2. Moshchuk, A., Bragin, T., Gribble, S.D., Levy, H.M.: A crawler-based study of spyware on the web. In: Proc. of the Annual Network and Distributed Systems Security Symposium (NDSS), San Diego, CA (February 2006)
3. Heron, S.: The rise and rise of the keyloggers. Network Security **6** (June 2007) 4–6
4. Cheswick, W.R., Bellovin, S.M., Rubin, A.D.: Firewalls and Internet Security: Repelling the Wily Hacker. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2003)
5. Federal Financial Institutions Examination Council: Authentication in an internet banking environment. http://www.ffiec.gov/pdf/authentication_guidance.pdf (2005) [Online; accessed on 10/2008].
6. Jagatic, T.N., Johnson, N.A., Jakobsson, M., Menczer, F.: Social phishing. Commun. ACM **50**(10) (2007) 94–100
7. Hoang, X., Hu, J.: New encryption model for secure e-commerce transactions using dsp—host, board and server communication issues-. In: Proceedings of the IEEE International Conference on Telecommunications. Volume 1. (2002) 166170
8. Hu, J., Xi, Z., Jennings, A., Lee, H.J., Wahyud, D.: Dsp application in e-commerce security. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). Volume II. (May 2001) 1005–1008
9. Weiss, K.P.: SecurID. RSA Security Inc. (1988) U.S. Patent 4720860.
10. Haller, N.: The s/key one-time password system. In: In Proceedings of the Internet Society Symposium on Network and Distributed Systems. (1994) 151–157
11. Iqbal, Z.: Secure mobile one time passwords for web services (master of science thesis). Technical report, Royal Institute of Technology (May 2006)
12. Hallsteinsen, S., Jorstad, I., Thanh, D.V.: Using the mobile phone as a security token for unified authentication. In: Proc. of the International Conference on Systems and Networks Communications (ICSNC), Washington, DC, USA, IEEE Computer Society (2007) 68
13. Me, G., Pirro, D., Sarrecchia, R.: A mobile based approach to strong authentication on web. In: Proc. of the International Multi-Conference on Computing in the Global Information Technology (ICCGI), Washington, DC, USA, IEEE Computer Society (2006) 67
14. Al-Qayedi, A., Adi, W., Zahro, A., Mabrouk, A.: Combined web/mobile authentication for secure web access control. IEEE Wireless Communications and Networking Conference (WCNC) **2** (March 2004) 677–681
15. Hager, C., Midkiff, S.: Demonstrating vulnerabilities in bluetooth security. Global Telecommunications Conference. IEEE GLOBECOM **3** (Dec. 2003) 1420–1424
16. Insight Consulting: How can bluetooth services and devices be effectively secured? Computer Fraud & Security (1) (Jan. 2006) 4–7
17. FreeAuth Project: The freeauth. http://www.freeauth.org [Online; accessed on 10/2008].
18. Mannan, M., van Oorschot, P.C.: Using a personal device to strengthen password authentication from an untrusted computer. In: Financial Cryptography (LNCS). Volume 4886. (2008) 88–103
19. JSR 177 Expert Group: Security and Trust Services API for Java^{TM}2 Platform, Micro Edition. http://jcp.org/aboutJava/communityprocess/final/jsr177/index.html (09 2004) [Online; accessed on 10/2008].
20. Open Base Movil Project: Openbasemovil. http://www.openbasemovil.org [Online; accessed on 10/2008].
21. The Legion of the Bouncy Castle: Bouncy castle lightweight crypto api. http://www.bouncycastle.org [Online; accessed on 10/2008].