On the Security of Today's Online Electronic Banking Systems

Joris Claessens, Valentin Dem, Danny De Cock, Bart Preneel and Joos Vandewalle

COmputer Security and Industrial Cryptography (COSIC), Dept. of Electrical Engineering – ESAT, K.U.Leuven, Kasteelpark Arenberg 10, B–3001, Leuven–Heverlee, Belgium

Abstract

Current technology is evolving fast and is constantly bringing new dimensions to our daily life. Electronic banking systems provide us with easy access to banking services. The interaction between user and bank has been substantially improved by deploying ATMs, phone banking, Internet banking, and more recently, mobile banking. This paper discusses the security of today's electronic banking systems. We focus on Internet and mobile banking and present an overview and evaluation of the techniques that are used in the current systems. The best practice is indicated, together with improvements for the future. The issues discussed in this paper are generally applicable in other electronic services such as E-commerce and E-government.

Keywords: E-banking, WWW security, WAP security

1 Introduction

Online electronic banking systems give everybody the opportunity for easy access to their banking activities. These banking activities may include: retrieving an account balance, money transfers between a user's accounts, from a user's account to someone else's account, retrieving an account history. Some banks also allow services such as stock market transactions, and the submission of standardized accounting payment files for bank transfers to third parties.

As technology evolves, different kinds of electronic banking systems emerge, each bringing a new dimension to the interaction between user and bank. The Automated Teller Machine (ATM) is the first well-known system that was introduced to facilitate the access of the user to his banking activities. Via a graphical user interface, the user can perform some of the transactions mentioned above. These are transmitted to the bank's computer system with which the device has established a communication link. The next step was the introduction of phone banking. Users can make a telephone call from home to the bank's computer system, and can use the phone's key pad to perform banking operations. The Internet offers a new alternative to the phone banking system. Via a more sophisticated and user-friendly interface, a browser or a dedicated standalone application, people can use the Internet to connect to the bank's computer system. Electronic devices are constantly getting smaller, while their functionality is extending. Now, mobile phones even offer the possibility to perform electronic banking.

1.1 Scope of the paper

This paper discusses the security of today's online banking systems. It concentrates on Internet, i.e., WWW[2], and mobile, i.e., WAP [32], banking. Our work is based on an extensive survey of the available online banking systems throughout the world. The survey was based on publicly available information. It was focused on the user's side of the system, and the interaction between the user and the bank. We investigated the security of the electronic banking systems of around 30 banks. This covered a small subset of the available online banking systems. However, we believe that this subset is representative with respect to the security offered by all current online banking systems.

1.2 Outline of the paper

The paper is organized as follows. In Section 2, we start by explaining the basic architectures of WWW and WAP electronic banking systems.

ELSEVIER ADVANCED TECHNOLOGY

Computers & Security
Vol 21, No 3, pp 257-269, 2002
Copyright ©2002 Elsevier Science Ltd
Printed in Great Britain
All rights reserved
0167-4048/02US$22.00

The main security requirements in an electronic banking system are identified. The important balance of security versus cost is shown. We briefly discuss the different services in an online banking system, which are related to the bank's internal structure. A distinction is then made between two important security aspects: while Section 3 discusses the security of the communication between client and bank, Section 4 elaborates on the additional client authentication issue. Additional security aspects are addressed in Section 5. Finally in Section 6, we formulate some concluding remarks. These include an indication of the current best practice, and some suggestions for future improvements.

2 Architecture and security requirements

2.1 Internet architecture

Figure 1 shows the basic architecture of an Internet electronic banking system. There are two participating entities: the user and the bank.

When the user has a PC with a network connection, the most common way to communicate with the bank is via a Web browser, hence banking through the World Wide Web [2] or Web banking. The standard protocol for communication between the browser and the bank's Web server is then used. It is often referred to as https, which is the http protocol on top of a security layer (see Section 3). Http is the communication language of the WWW.

A bank will mostly require more security functionality than an ordinary browser is able to provide. Historically, due to US export restrictions, this extra security functionality especially included strong cryptography (see Section 3.6).

A dedicated standalone client/server application is therefore an alternative way to realize communication between the user and the bank. The same protocol as used by the Web browser/server can be used here to provide security. However, the bank must provide the user with the necessary software, as the electronic banking system does not rely on the browser that is already installed on the user's computer.

To avoid the problem of distribution and installation of extra software on the user's computer, banks often deploy an intermediate solution. An ordinary browser is used at the client side, but to increase the functionality, a Java applet is downloaded from the bank's website. This applet is a relatively small piece of software code that runs within the user's

Figure 1: Internet banking structure.

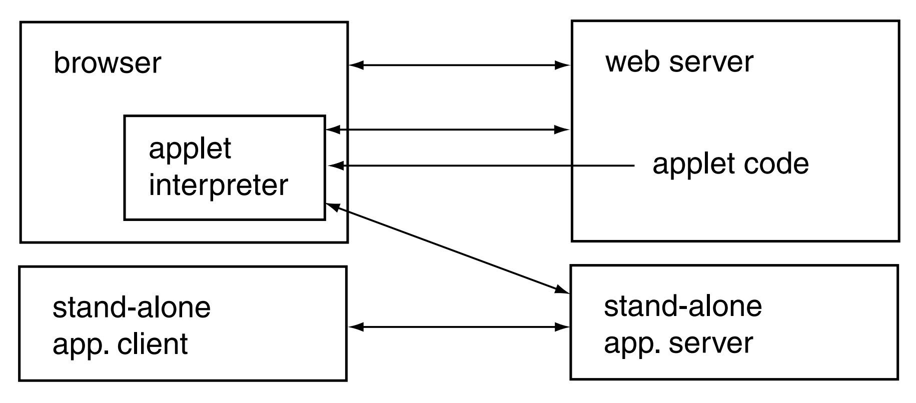
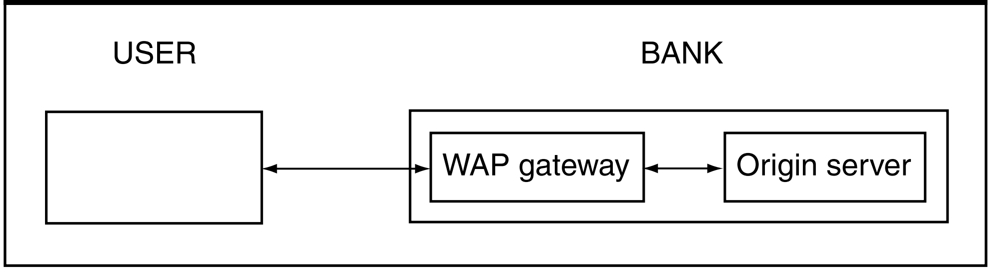


Figure 2: WAP banking structure.



browser, and that will provide extra security functionality. A big advantage of this approach is that the applet technology allows the bank to easily maintain and update the client software. Clients will automatically download and use new versions of the software. Banks do not need to distribute new software in an old-fashioned way. Note that in the scope of this paper we intend to use the term 'user' when we want to refer to a physical person. With the term 'client' we want to refer to the machine and the software, as in the traditional client/server sense.

2.2 WAP architecture

Figure 2 shows the basic architecture of an electronic banking system using the Wireless Application Protocol (WAP) [32]. There are again two participating entities: the user and the bank.

When the user has a WAP-enabled mobile phone, he can connect to the bank via a wireless link. WAP is essentially a wireless equivalent to the Internet protocol stack (TCP/IP). To connect the wireless domain to the Internet, a WAP proxy or gateway is needed to translate the protocols used in WAP to the protocols used on the Internet. For example, the WAP proxy encodes/decodes the content to reduce the size of the data that has to be sent over the wireless link.

Just as for the WWW, the user interface to the electronic banking application is via a mini browser in the mobile phone. The communication between the mobile phone and the WAP proxy is secured with a protocol that is very similar to the one used in the Internet case (see Section 3).

A major difference in a WAP banking system is that there is no *end-to-end* connection between the client and the bank's (Web) server. Banks should therefore not rely on the client's default (and untrusted) gateway, but have their own trusted and secure gateway within a secure environment together with their server.[1]

2.3 Security requirements

The following general security requirements [22] also apply to electronic banking systems:

- *Confidentiality* ensures that only authorized entities have access to the content of the exchanged information. E.g., an eavesdropper should not be able to find out what transactions a particular user is executing.
- *Entity authentication:* users should be sure that they are communicating with the real bank, before sending sensitive information to it; banks should know the identity of a user before processing its transactions.
- *Data authentication* – i.e., *data origin authentication* and *data integrity* – allows one to detect manipulation and replay of data, by unauthorized parties; data manipulation includes insertion, deletion and substitution. E.g., users and the bank want

[1] *WAP is currently often deployed over the bearer GSM – Global System for Mobile communications [31]. Switching from the gateway of the client's provider, to the bank's gateway can then for example be done by sending a Short Message Service (SMS) message.*

to be sure that the information they receive is genuine and fresh (not replayed).

- *Non-repudiation* prevents an entity from denying previous commitments or actions. E.g., a bank should be able to prove to a third party that a user performed a certain transaction, in case that user denies having performed it.

2.4 Cost versus security

Security measures are needed to prevent certain risks and associated costs to occur. The security measures, however, also impose a certain cost by themselves. A proper balance should therefore be made between the investments in security measures and the potential costs that a bank might have to cope with due to remaining risks without these measures. Particularly in electronic banking systems, the extra cost at the client side is reduced as much as possible. Users should be able to perform electronic banking with the standard infrastructure and software that is already available. This makes the electronic banking service more attractive, but might unfortunately have an impact on the security level this service can offer. In practice, banks try to have a minimal level of security alleviating most of the risks, with maximum level of convenience.

2.5 Services in an electronic banking system

This paper focuses on the user's side and the interaction between the user and the bank. Nevertheless, we believe that we can provide additional insight in an electronic banking system if we also briefly discuss the bank's internal structure.

Let us consider a fictitious electronic banking system. Four major functionalities are present in our fictitious bank. As 'separation of concerns' increases the modularity, maintainability, etc., and as such security, these functionalities are implemented as four distinct services:

(1) To interact with clients, the bank requires an *Interface Server*. An interface is needed for all different environments: ATM, WWW, WAP.

(2) The bank should verify the authenticity of client requests and should authenticate its responses. This is done by the *Authentication Server*.

(3) Financial transactions are validated by the *Transaction Server*. This server also provides an easy interface towards the mainframe.

(4) Finally, the Mainframe executes the transactions and keeps the financial records.

This structure reflects the different services that are present in any electronic banking system. The security of the communication between the client and the bank (as discussed in Section 3) is handled in the Interface Server part. The authentication of the user (as discussed in Section 4) is handled by the Authentication Server.

Note that it would be possible to rely on a third party to provide for example the authentication service. Microsoft's Passport [28] is such a service that is intended to provide a single identity with which a user could perform all its online activities. Although this might work for some E-commerce merchants, authentication in an electronic banking application is far too critical to entrust it to another party.

3 Communications security

In this section the security of the communication between the client and the bank is discussed. We restrict ourselves to the standard solutions that are used within almost all electronic banking systems, namely SSL/TLS and WTLS, for Internet and WAP banking respectively.

3.1 SSL/TLS/WTLS in general

Secure Sockets Layer (SSL) was originally an initiative of Netscape. The IETF adopted SSL for its Transport Layer Security (TLS) protocol [4]. The WAP Forum adapted TLS to create a wireless equivalent, Wireless Transport Layer Security (WTLS) [33]. Detailed background information on SSL/TLS can be found in Rescorla [25]. Although there are differences between (several versions of) these protocols, conceptually they provide the same security service: a secure channel between client and bank.

3.1.1 Secure channel

SSL/TLS/WTLS provides a secure communication channel between the client and the bank. This means that the data that is transmitted between both ends is kept secret (confidentiality) and that tampering will be detected (data integrity); the bank is always authenticated; the client can be required to authenticate too. Note that many electronic banking systems do not rely on the client authentication feature of the secure channel, but rather implement a client authentication mechanism on top of this channel (see Section 4).

An advantage of SSL/TLS/WTLS is that it can easily be used underneath various communication protocols, including but not restricted to http. As SSL/TLS/WTLS only provides a secure channel, it does not provide non-repudiation: at the receiving side, the transmitted data leaves the secure channel, and the cryptographic protection is removed; there is no digital signature on client data. Electronic banking systems should therefore implement a non-repudiation mechanism on top of the secure channel. However, note that this is rarely done in practice.

3.1.2 Handshake and data transfer

A connection between the client and the bank is divided into two phases, the *handshake* and the *data transfer*.

The purpose of the *handshake* is three-fold: client and bank need to agree on a set of cryptographic algorithms that will be used to protect the data, to authenticate each other, and to agree on cryptographic keys; secondly, they need to establish a set of cryptographic keys with which data will be protected; lastly, the bank authenticates to the client and, optionally, the client authenticates to the bank.

Once the handshake has been completed, *data transfer* can take place. Data is broken up in fragments, and transmitted as a series of protected records. To provide data integrity, a Message Authentication Code (MAC) is computed over a data fragment; fragment and MAC are then encrypted.

3.2 SSLv2/SSLv3/TLS

The first public version of SSL, version 2, suffered from a number of security flaws, which have been fixed in SSLv3. As browsers nowadays still support SSLv2, and as it is still in use in some systems, we briefly sum up its security problems [25]. The same cryptographic keys are used for message authentication and for encryption; this means that in export mode the security of the MACs is unnecessarily weakened. SSLv2 has a weak MAC construction and relies solely on the MD5 hash function; Dobbertin has demonstrated the vulnerabilities of MD5 in [6]. SSLv2 does not have any protection for the handshake, hence a person-in-the-middle attack cannot be detected. Finally, a truncation attack is possible, as SSLv2 simply uses the TCP connection close to indicate the end of data, so that the attacker can simply forge the TCP FINs and the recipient cannot tell that it is not a legitimate end of data.

The IETF TLS working group adopted the SSLv3 protocol. Some minor modifications were made to increase the security: the way cryptographic keys are expanded from the initially exchanged secret was improved, the HMAC-like MAC construction was replaced by the real HMAC [18] and, implementations were required to include support for the Diffie-Hellman key agreement [24], the Digital Signature Standard [10], and Triple-DES [9] encryption. The IETF TLS working group is at the time of writing of this article working on several enhancements to the TLS protocol, including (wireless) extensions, and new ciphersuites incorporating Rijndael [3], the Advanced Encryption Standard [11].

3.3 WTLS

The WAP Forum has adapted TLS to make it suitable for a wireless environment with small devices, which have limitations on bandwidth, memory and processing. WTLS therefore includes the usage of elliptic curve cryptography (limited memory and processing) by default; WTLS does also work on top of a datagram instead of a connection based communication layer (compare to UDP vs. TCP on the Internet); finally, WTLS defines its own certificate format optimized for size (limited bandwidth), but supports the ordinary X.509 certificate (see Section 3.5) too.

3.4 Implementation issues

Today's popular browsers implement the SSL/TLS protocol by default.[2] As already indicated, when presenting the basic architecture of an Internet banking system, many banks prefer to use their own (or a third party's) implementation of SSL/TLS, executed from within the browser or via a standalone application. The main reason behind this was the existence of the US export restrictions (see Section 3.6).

As an alternative to the browser-based electronic banking application, a standalone application can be used which provides both the necessary security and the banking functionality. Sometimes a standalone application is only used as a proxy running on the local machine of the user, that sits in between the browser and the bank's server, and adds strong security for the communication between the client and the bank (while the communication between the browser and the proxy remains weak). In many cases, program code including strong security and possibly banking functionality is just downloaded as an applet, and runs within the browser. This applet can again have complete functionality, or behave as a proxy in between the browser and the bank's server.

3.5 Trust anchors

The user can only trust the correct execution and interface of the electronic banking application, if he has a genuine copy of the browser or the standalone application. If a Java applet is used, this should be digitally signed with the bank's private signing key,[3] so that it can be verified by the user before actually executing it. Users must recognize when they have a secure session with the bank. However, in today's browsers, there are only some limited visual indications (e.g., closed lock), and an inexperienced user is easily fooled by a spoofed website, as demonstrated by Felten et al. [7] and more recently by Yuan et al. [34].

Above all, there must exist a meaningful Public Key Infrastructure (PKI). A PKI ensures a correct mapping of public keys to entities. An entity's name together with the corresponding public key is put into an X.509 [14] certificate which is signed by a Certification Authority (CA). The authentic distribution of 'root' certificates, the certificates of the CAs, is here very important. They are used by the bank to verify entity certificates (if client authentication is based on this mechanism; see Section 4). The client needs them in order to verify the bank's certificate during SSL/TLS

[2] *Netscape 4.7x only supports SSL, while Netscape 6.x and Microsoft Internet Explorer 5 and 6 support both SSL and TLS.*

[3] *See Microsoft's 'Authenticode' or Netscape's 'Object Signing' [21].*

authentication, and in order to verify digitally signed applets. Authentication cannot be performed without authentic root certificates. Root certificates are usually included in the installation of today's Web browsers. If the electronic banking system is based on a standalone application, a root certificate is usually hard-coded in the software, or put in a configuration file. In all cases, they are typically only protected by the operating system. Hence, if the root certificates on a user's machine are replaced by fake ones, an attacker can impersonate the bank, without the client noticing it.

3.6 Export restrictions

Historically, the US export restrictions prevented common browsers (i.e., those originating from the US) to contain (activated)[4] strong cryptography. For an exported US browser, the length of a symmetric cryptographic key used to be limited to 40 bits; asymmetric keys (RSA) limited to 512 bits.

During the relaxation of these restrictions, financial institutions were allowed to enable the strong cryptography which was already present in browsers. This was technically done through the use of special server certificates with particular extensions. Different terminology was used: Microsoft's "Server Gated Cryptography" (SGC), Netscape's International "Step-Up" encryption and Verisign's "Global Server ID".

Since the beginning of 2000, cryptographic software can be exported from the US to most other countries without any restrictions. However, the proprietary implementations of SSL/TLS external to the browser are still frequently used (one out of three banks in our survey do not rely on the browser's implementation of SSL/TLS).

[4] *Exported browsers, however, did contain strong cryptography. For Netscape, a small program was made available on the Internet to activate this strong crypto: http://www.fortify.net/.*

4 Client authentication

Providing a secure communications channel from a client to an authenticated bank is only part of a secure electronic banking system. Authenticating the client is the other crucial part.

4.1 Entity vs. transaction authentication

An important distinction should be made between entity and transaction authentication. Entity authentication means that the client authenticates when initiating a session with the bank. Transaction authentication means that individual transactions within this session are authenticated by the client. Depending on the authentication mechanism, transaction authentication can provide non-repudiation of single transactions, while entity authentication does not provide non-repudiation of transactions. This distinction is clearly made in today's electronic banking systems. In some systems only entity authentication is present, while other systems incorporate transaction authentication too. Both entity and transaction authentication can be provided in various ways, as presented in the remainder of this section.

4.2 Authentication mechanisms

4.2.1 Fixed passwords

Many electronic banking systems still rely on a fixed password to authenticate the client (two out of three in our survey). This password can be a PIN number or a character-based password, and is often combined with a service account number that is not easy to guess (i.e., unrelated to the user's name or bank account). In many cases only a subset of the digits has to be provided by the user, which provides some security against an attacker who is looking over the shoulder or is sniffing the keyboard (the bank asks for a different subset each time).

Fixed passwords are very often used for entity authentication. In rare cases they are also used for transaction authentication, although this is rather entity authentication for a single-transaction session. In many systems fixed passwords are used for entity authentication and combined with one of the mechanisms explained in the following sections, used for transaction authentication. Moreover, a password is almost always required to bootstrap the system the first time the user starts using it. This is typically different from the password(s) used in the normal sessions.

Passwords should never be sent in clear over the network. They are, however, also vulnerable to dictionary attacks, password guessing and social engineering. Although these risks have been known for a long time [23], fixed passwords are still widely used, for they are very easy to implement and use.

4.2.2 Dynamic passwords

Banks sometimes issue a list of one-time passwords to their users (at least two banks in our survey). These can only be used once, and should therefore offer more security. However, it is very difficult to learn these by heart, and so the users will be forced to keep a list somewhere either on paper, or worse, on a file on their PC. Terminology banks use includes 'scratch list number', mostly used for entity authentication, and 'transaction number', used to authenticate individual transactions. Some systems use a combination of fixed and dynamic passwords: fixed for entity authentication and dynamic for transaction authentication. Instead of issuing a list of independent passwords, it is possible to generate a chain of dependent one-time passwords; see for example the system described by Haller et al. in [13] which is based on an idea described by Leslie Lamport in [19]. This usually requires extra software at the client side.

4.2.3 Challenge/response

The idea of a challenge/response scheme is that the client proves his identity to the bank (i.e., entity authentication) by demonstrating knowledge of a secret, not by just sending this secret to the bank, but by producing the proper response to a random challenge using this secret. There are symmetric and asymmetric challenge/response schemes. In a symmetric scheme for example, the response consists of a MAC on the time or on a random challenge of the bank. A digital signature on a random challenge message is an example of an asymmetric scheme. These challenge/response schemes are often implemented with hardware tokens.

4.2.4 SSL/TLS/WTLS

Using a digital signature for a challenge/response scheme is actually an option in the SSL/TLS/WTLS protocols. When setting up a secure channel between the client and the bank, the client can also be authenticated explicitly using a digital signature: during the handshake, the client signs a hash of all the previously exchanged handshake messages. Usually, the private signing key is stored on the user's computer and is only protected with a password (see 4.2.5). Note that due to constraints in current WAP phones, this is not yet practically used in WAP banking systems.

4.2.5 Digital signatures

Besides entity authentication, digital signatures can also be used for transaction authentication. This is the most secure alternative. However, of today's browsers, only Netscape includes a Javascript mechanism to digitally sign, for example, the contents of a form. So electronic banking systems normally use their own implementation, a standalone application or an applet, for digital signatures (one bank out of seven in our survey).

As indicated before, the private signing key is usually stored on the user's computer and is only protected with a password. Moreover, Shamir and van Someren [27] have shown that cryptographic keys are very vulnerable in software. This has been successfully verified by

Janssens et al. in [16]. Still, the level of security is substantially higher than when using fixed passwords.

4.2.6 Hardware tokens

Several of the previously discussed mechanisms can be (more securely) implemented using hardware tokens (one bank out of 10 in our survey).

Private digital signature keys for transaction authentication can be kept on smart cards. Due to the cost of issuing extra smart cards to users, this highly secure solution is not frequently deployed. However, existing smart card applications can be and are used as a means for entity authentication, e.g., electronic purses, or an electronic identity card.

Note that one can never achieve perfect security, but only (substantially) increase the cost required for a successful attack. As such, more advanced attacks on smart cards remain possible, as shown by Anderson and Kuhn in [1] and Kocher et al. in [17].

Hardware tokens which display a response to the current time interval (e.g., SecurID [26]) or to an unpredictable challenge given by the bank via the computer screen and typed in by the user on the token (e.g., Digipass [5]), are used for entity authentication. These hardware tokens can sometimes also calculate MACs for the purpose of transaction authentication.

Mobile devices such as PDAs or special-purpose wireless wallets can enhance an Internet or WAP banking system's security. These devices are considered personal, and can perform cryptographic protocols, to provide both entity and transaction authentication. The communication between the device and the PC or WAP phone is realized manually, with Bluetooth [15] or with an infrared interface.

The compromise of one token could lead to the disastrous scenario in which the bank should issue a new token to all users. To prevent this, all tokens should contain a different cryptographic key. When using asymmetric keys this is not a problem. However, in the case of symmetric keys, it requires the maintenance of a secure database containing all the secret keys the bank shares with its users. Symmetric keys are therefore often cryptographically derived from a unique serial number of the token and a master key that is the same for all (or a subset of all) the tokens. In this way, each user shares a different key with the bank, without the problem of the secure database.

5 Additional security issues

Communications security, i.e., a secure channel between a client and an authenticated bank, and client authentication, i.e., entity and transaction authentication, are the two main security issues present in an electronic banking system. There are, however, some additional security issues, which unfortunately in practice are sometimes the most critical ones.

5.1 Registration

Before a user can actually use a secure electronic banking system, a registration procedure is performed. During this registration procedure, the security of the system is bootstrapped: the user has to obtain an initial means for entity authentication, with which a first secure session can be established with the bank; thereafter, the regular security authenticators are enabled. Usually an initial password is used which the user obtained via paper mail, and/or after having physically authenticated in one of the bank's offices. Sometimes, user authentication via the phone is required at the first login at the bank's system; the user then for example has to give the operator a challenge displayed on screen, and/or answer some questions. It is clear that if something goes wrong during this stage, the security of the rest of the electronic banking system is undermined.

5.2 Delegation

In some situations, the ability of delegation within an electronic banking system is desired. For example, the manager of a department would like to be able to delegate (restricted) rights to one or more employees who need to have access to the company's banking transactions, or possibly parents would like to delegate certain rights to their children.

If the authentication of the client is only based on a fixed password, delegation can unfortunately be performed by just sharing the password. However, this is not delegation but rather impersonation. If client authentication is more secure, it is more difficult to give away the necessary secret information, or at least, it is difficult to duplicate it, for example in the case of smart cards.

A small number of existing electronic banking systems implement some kind of real delegation mechanism. For example, the owner of a bank account can create additional password-protected accounts with limited rights.

5.3 Secure platforms

When discussing the establishment of a secure communications channel between a client and the authenticated bank, and the authentication of the client, the assumption has to be made that the user's PC, operating system and software form a secure end-point in this process. This is unfortunately mostly not true in reality. Critical trusted anchors (see earlier) are thus not always present. Typical client platforms have shown to be very vulnerable, as described by Loscocco et al. [20]. Viruses, Trojan horses, worms, and other malicious programs can tamper with the installed root certificates, they can steal a user's private keys, they can spoof the user interface or mislead users in another way, they can intercept communication before it is 'securely' sent to the bank,[5] etc.

Even smart cards may not protect against this problem. If the card is unlocked by typing a PIN code via the ordinary keyboard, the PIN code could be captured or the user could be requested for the PIN code through a fake interface. Ideally, a smart card reader should therefore have its own pinpad and a small display. Users do not have to enter PIN codes via their computer's keyboard then. Users are then also able to verify crucial information on a trusted display, i.e., not their regular computer screen. Common specifications for such secure readers are being developed [8]. However, it still constitutes an expensive solution. Most current smart card readers therefore only consist of a slot in which the card can be inserted. This still protects the card's private key itself, but does in theory not prevent access to the card's signing function.

An industry alliance is currently working on mechanisms that will provide more trust and enable security on an end-user's computing platform [30]. This effort will in particular enable a more secure creation of digital signatures on an end-user's computing platform [29].

Also the bank's server should form a secure end-point in the electronic banking system. The appropriate measures should be taken to prevent hackers from breaking into the site. A discussion of these measures falls out of the scope of this article, but detailed information can be found in Garfinkel and Spafford [12].

[5] *For example, if the electronic banking system includes a proxy on the user's machine that provides the end-point of a secure channel to the bank and that sits in between the user's browser and the bank's Web server, a malicious program could sniff the communication between the browser and that proxy.*

5.4 The human factor

The fact that a client platform is not secure is often just due to the lack of security knowledge of the end-user. Although the typical client platform is inherently insecure, most problems could be prevented by an educated, careful and security-conscious user. Users should keep their security authenticators, whether these are just passwords, a list of one-time passwords, hardware tokens, or the PINs to unlock these tokens, private, and protect them from potential abuse. Users should install virus scanners, and keep them and their system up-to-date. Users should avoid practices that easily lead to security hazards, in particular they should not start up arbitrary executable attachments received via electronic email. Users should check fingerprints of certificates against the fingerprints that are (should be) given by the bank on official paper documents. Banks should provide information to their users about these matters. In fact, they will mostly do this to be able to decline responsibility in case something goes wrong and it turns out it is due to the user. The bank's system administrators should be properly trained in computer security. They should for example regularly monitor security advisories and apply software patches when required.

5.5 Logging and monitoring

The previous sections made clear that there is no such thing as perfect security. No matter the amount and strength of security measures in place, there will always be remaining potential security weaknesses. As some security breaches cannot be completely prevented (e.g., obtaining and misusing a user's credentials), logging and monitoring will allow the bank to at least detect security hazards, or find out later what exactly happened. These detection mechanisms can go from just passive logging to active monitoring, e.g., sending an alert to the bank if certain transactions do not match a user's regular profile.

6 Concluding remarks

As technology evolves, more of our daily life's activities are moved online. Electronic banking is an important example of this trend. The issues discussed in this paper are generally applicable to E-commerce, E-government, and other E-services. A first main security issue consists of the establishment of a secure channel to provide data confidentiality and data integrity of communications between a client and an authenticated bank. The second security issue is the authentication of the client, at the beginning of a session, i.e., entity authentication, and for each transaction, i.e., transaction authentication.

With respect to the first issue, almost all today's electronic banking systems rely on the SSL/TLS/WTLS protocols. Although similar solutions would be possible, this is certainly the most popular alternative, as it is available in all common Web browsers. From a cryptographic point of view, SSL/TLS/WTLS has proven to be a practically secure protocol. With respect to the second issue, fixed passwords are still most widely used, as they are easy to implement and use. From a security point of view, public-key cryptography is the best solution, and the only one providing non-repudiation. However, the client's private key is in practice often stored in software and only protected by a password. Smart cards (and readers) constitute a solution that still seems to be too expensive or complicated. The best current practice – i.e., with proper balance between security and cost – therefore is probably the use of other hardware tokens, such as a Digipass, that generate responses to unpredictable challenges of the bank and that are able to calculate MACs, or tokens such as a smart card that is already used in other (related) applications, as in, for example, an electronic purse.

Even the best solution relies on the assumption that the end-points of the system are trusted. Concerning the client, this means that the electronic banking system should run within a trusted environment that has not been tampered with. This is, however, far from trivial in a typical client environment. The user must be responsible for maintaining his computing environment, and keeping it trustworthy. The user must also protect the secret elements that are used in the system, such as passwords, private keys, or tokens. While inherently more secure environments will considerably improve the overall security of an electronic banking system, education of users has an even more significant impact on its overall security. Banks in their turn should ensure the security of their servers.

Security is all about risks and associated costs. In an electronic banking system, the cost of security measures at the client side is reduced as much as possible, while keeping a minimal level of protection. One cannot achieve perfect security, so at some moment in time, despite all security measures something will always go wrong. The real question is what will happen in that case. Banks of course will want to pass liability to their users and vice versa. Law and small print on contracts will somehow make clear who will have to take the risk in the end.

Acknowledgements

The authors would like to thank the anonymous referees for their helpful comments.

References

[1] *Ross Anderson and Markus Kuhn. Tamper Resistance – a Cautionary Note. In Proceedings of the Second USENIX Workshop on Electronic Commerce, 1996, pp. 1–11.*

[2] *Tim Berners-Lee and Mark Fischetti. Weaving the Web – The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor. HarperCollins Publishers, 1999.*

[3] *Joan Daemen and Vincent Rijmen. Rijndael, the Advanced Encryption Standard. Dr. Dobb's Journal, 26(3), March 2001, pp. 137–139.*

[4] *Tim Dierks and Christopher Allen. The TLS Protocol Version 1.0. IETF Request for Comments, RFC 2246, January 1999.*

[5] *Digipass. http://www.vasco.com/.*

[6] *Hans Dobbertin. The Status of MD5 After a Recent Attack. RSA Laboratories' CryptoBytes, 2(2), 1996, pp. 1–6.*

[7] *Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach. Web Spoofing: An Internet Con Game. In Proceedings of the 20th National Information Systems Security Conference, 1997, pp. 95–103.*

[8] *FINREAD. Financial Transactional IC Card Reader. CEN Workshop Agreement, CWA 14174, July 2001.*

[9] *Federal Information Processing Standard. Data Encryption Standard (DES). FIPS PUB 46-3, US Department of Commerce, National Institute of Standards and Technology, October 1999.*

[10] *Federal Information Processing Standard. Digital Signature Standard (DSS). FIPS PUB 186-2, US Department of Commerce, National Institute of Standards and Technology, January 2000.*

[11] *Federal Information Processing Standard. Advanced Encryption Standard (AES). FIPS PUB 197, National Institute of Standards and Technology, November 2001.*

[12] *Simson Garfinkel and Gene Spafford. Practical UNIX & Internet Security, 2nd Edition. O'Reilly, 1996.*

[13] *Neil Haller, Craig Metz, Phil Nesser, and Mike Straw. A One-Time Password System. IETF Request for Comments, RFC 2289, February 1998.*

[14] *ITU-T Recommendation X.509 – ISO/IEC 9594-8. Information Technology – Open Systems Interconnection – The Directory: Public-Key and Attribute Certificate Frameworks. 4th edition, 2001.*

[15] *Markus Jakobsson and Susanne Wetzel. Security Weaknesses in Bluetooth. In D. Naccache, editor, Topics in Cryptology – Proceedings of the Cryptographers' Track at RSA 2001, Lecture Notes in Computer Science, LNCS 2020, Springer-Verlag, 2001, pp. 176-191.*

[16] *Dirk Janssens, Ronny Bjones, and Joris Claessens. KeyGrab TOO – The search for keys continues... Utimaco White Paper, 2000. 11 pps. http://www.utimaco.com/.*

[17] *Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In M. Wiener, editor, Advances in Cryptology – CRYPTO'99, Lecture Notes in Computer Science, LNCS 1666, Springer-Verlag, 1999, pp. 388-397.*

[18] *Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-Hashing for Message Authentication. IETF Request for Comments, RFC 2104, February 1997.*

[19] *Leslie Lamport. Password Authentication with Insecure Communication. Communications of the ACM, 24(11), November 1981, pp. 770–772.*

[20] *Peter A. Loscocco, Stephen D. Smalley, Patrick A. Muckelbauer, and Ruth C. Taylor. The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments. In Proceedings of the 21st National Information Systems Security Conference, October 1998, pp. 303–314.*

[21] *Gary McGraw and Ed Felten. Security JAVA – Getting Down to Business with Mobile Code. John Wiley & Sons, 1999.*

[22] *Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996.*

[23] *Robert Morris and Ken Thompson. Password Security: A Case History. Communications of the ACM, 22(11), 1979, pp. 594–597.*

[24] Eric Rescorla. Diffie-Hellman Key Agreement Method. IETF Request for Comments, RFC 2631, June 1999.

[25] Eric Rescorla. SSL and TLS: Designing and Building Secure Systems. Addison-Wesley, 2000.

[26] SecurID. http://www.rsasecurity.com/.

[27] Adi Shamir and Nicko van Someren. Playing "Hide and Seek" with Stored Keys. In M. Franklin, editor, Proceedings of the Third International Conference on Financial Cryptography, Lecture Notes in Computer Science, LNCS 1648, Springer-Verlag, 1999, pp. 118-124.

[28] Marc Slemko. Microsoft passport to trouble. http://alive.znep.com/ ˜marcs/passport/, November 2001.

[29] Adrian Spalka, Armin B. Cremers, and Hanno Langweg. Protecting the Creation of Digital Signatures with Trusted Computing Platform Technology Against Attacks by Trojan Horse Programs. In Michel Dupuy and Pierre Paradinas, editors, Trusted Information – The New Decade Challenge – Proceedings of IFIP SEC 2001, Kluwer Academic Publishers, 2001, pp. 403-419.

[30] TCPA. Trusted Computing Platform Alliance. http://www.trustedpc.org/.

[31] Klaus Vedder. GSM: Security, Services and the SIM. In B. Preneel and V. Rijmen, editors, State of the Art in Applied Cryptography, Lecture Notes in Computer Science, LNCS 1528, Springer-Verlag, 1998, pp. 227-243.

[32] Wireless Application Protocol Forum. http://www.wapforum.org/.

[33] Wireless Application Protocol Forum. WAP Wireless Transport Layer Security. Version 06-Apr-2001.

[34] Yougu Yuan, Eileen Zishuang Ye, and Sean Smith. Web Spoofing 2001. Department of Computer Science, Dartmouth College, Technical Report TR2001-409, July 2001.