

Secure Internet Banking Authentication

The authors present two challenge–response Internet banking authentication solutions—one based on short-time passwords and one on certificates—and then describe how easily these solutions can be extended should sophisticated content-manipulation attacks arise.

ALAIN HILTGEN
UBS Wealth Management and Business Banking, Zurich

THORSTEN KRAMP AND THOMAS WEIGOLD
IBM Zurich Research Laboratory

The Internet is an integral part of our daily lives, and the proportion of people who expect to be able to manage their bank accounts anywhere, anytime is constantly growing. As such, Internet banking has come of age as a crucial component of any financial institution's multichannel strategy.

Information about financial institutions, their customers, and their transactions is, by necessity, extremely sensitive; thus, doing such business via a public network introduces new challenges for security and trustworthiness. Any Internet banking system must solve the issues of authentication, confidentiality, integrity, and nonrepudiation, which means it must ensure that only qualified people can access an Internet banking account, that the information viewed remains private and can't be modified by third parties, and that any transactions made are traceable and verifiable. For confidentiality and integrity, Secure Sockets Layer/Transport Layer Security (SSL/TLS) is the de facto Internet banking standard, whereas for authentication and nonrepudiation, no single scheme has become predominant yet.

This article describes current authentication threats and two proposed solutions as well as how these solutions can be extended in the face of more complex future attacks. This work falls in line with recent federal regulator initiatives (see www.ffiec.gov/press/pr101205.htm and www.info.gov.hk/hkma/eng/press/2005/20050530e3_index.htm) requiring, or at least expecting, banks to rapidly adopt two-factor authentication for Internet banking.

Attacks on authentication

Internet banking systems must authenticate users before granting them access to particular services. More pre-

cisely, the banking system must determine whether a user is, in fact, who he or she claims to be by asking for direct or indirect proof of knowledge about some sort of secret or credential. With the assumption that only an authentic user can provide such answers, successful authentication eventually enables users to access their private information.

We can classify all Internet banking authentication methods according to their resistance to two types of common attacks: offline credential-stealing attacks (see Figure 1) and online channel-breaking attacks (see Figure 2).

Offline credential-stealing attacks aim to fraudulently gather a user's credentials either by invading an insufficiently protected client PC via malicious software (such as a virus or Trojan horse) or by tricking a user into voluntarily revealing his or her credentials via phishing. Several security precautions can help users protect themselves from malicious software—for example, installing and maintaining a firewall and up-to-date antivirus software, regularly applying operating system and browser patches, and configuring security software appropriately—but most users don't always follow these precautions. Phishing, in contrast, works by hijacking the trusted brands of well-known financial institutions and tricking users into entering their credentials in fake Web forms. Phishing can also be combined with pharming, an exploitation of a DNS server software vulnerability that redirects a Web site's traffic to a fake site. Common sense aside, phishers still manage to convince up to 5 percent of a spoofed email message's recipients to reveal their personal information (www.antiphishing.org/APWG_Phishing_Ac-

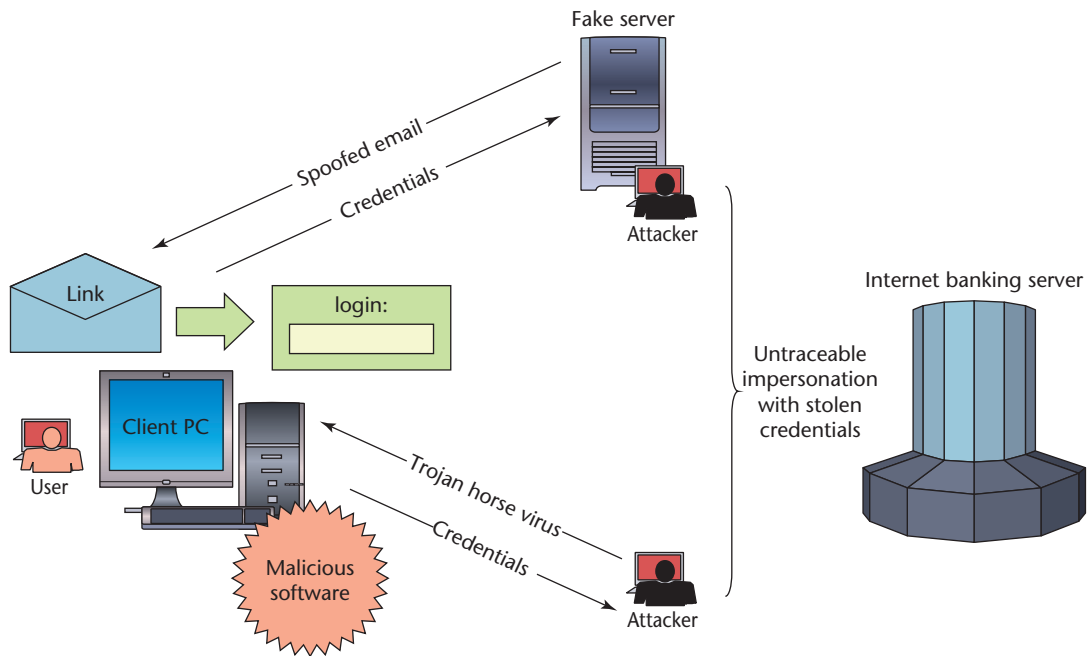


Figure 1. Offline credential-stealing attack scenarios. User credentials are attacked via interception on the client PC or by tricking the user into revealing them to a fake server.

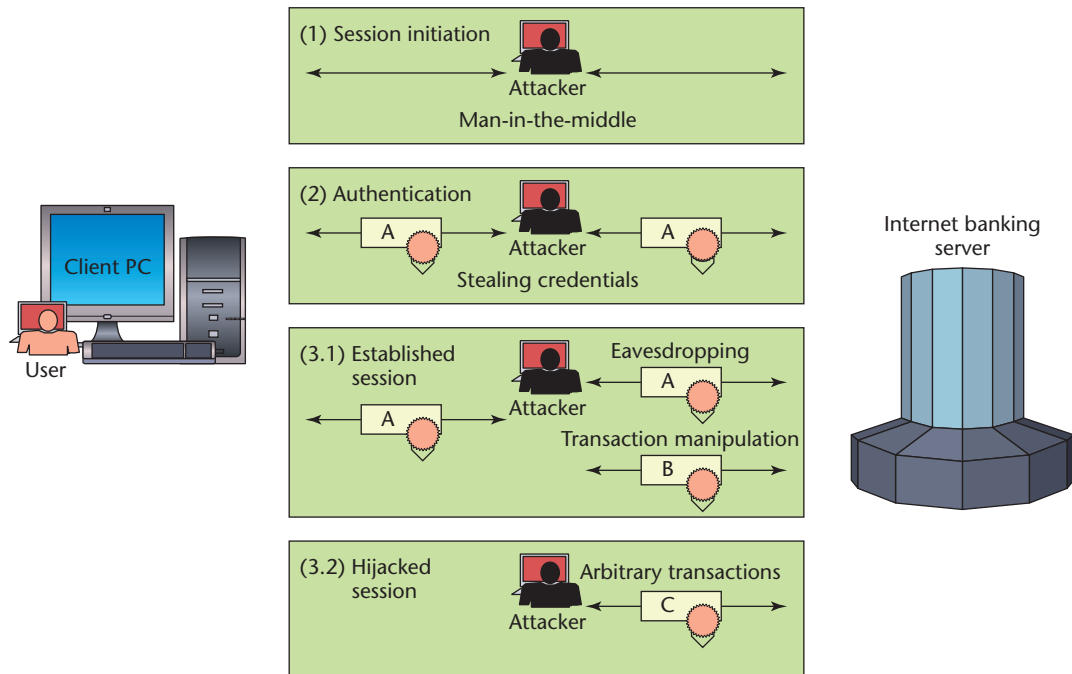


Figure 2. Online channel-breaking attack scenarios. Session credentials (such as session cookies) are attacked via interception as they move between the client PC and the banking server.

tivity_ReportOct2004.pdf). This success rate is at least partially because most users don't actually know how

to reliably identify a genuine banking server or are fooled by a fake server that online reproduces parts of a

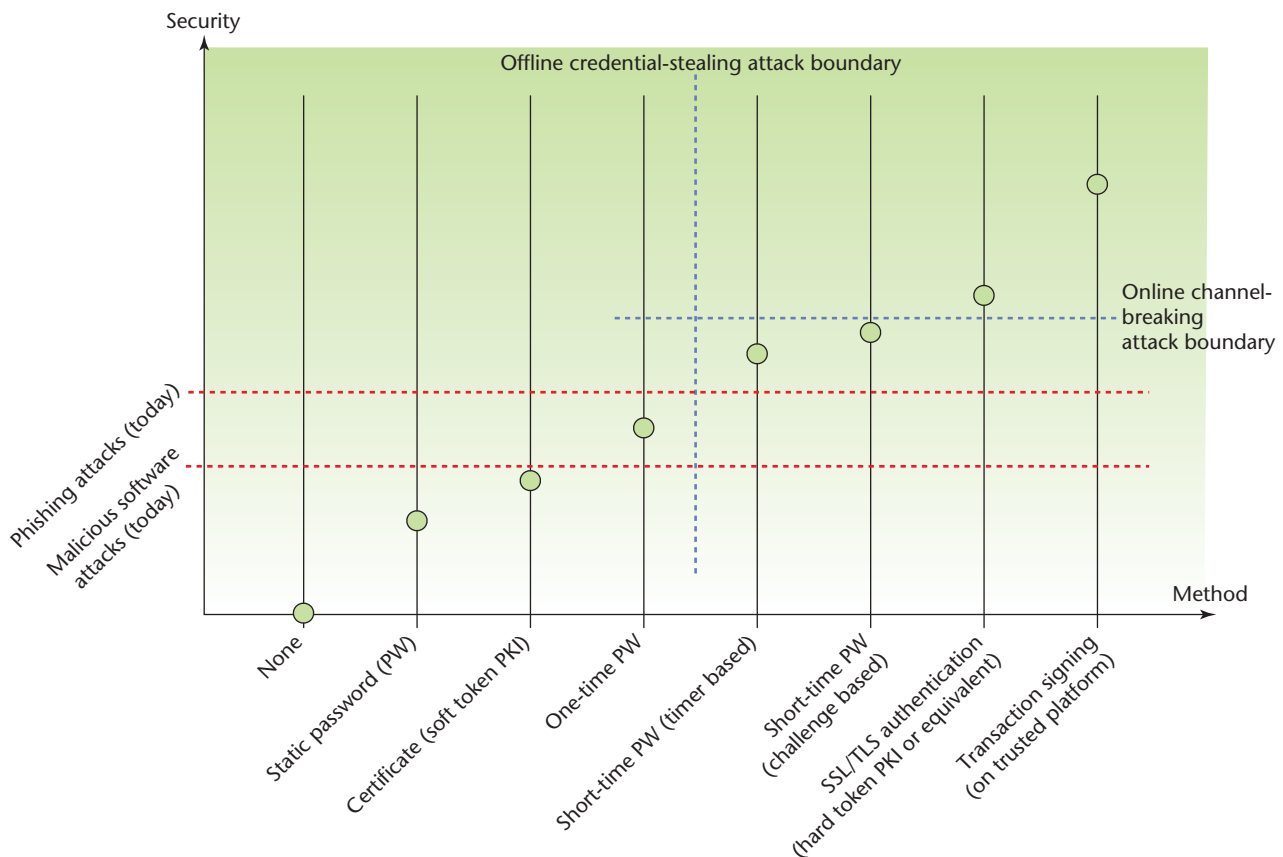


Figure 3. Taxonomy of Internet banking authentication methods. Methods are classified according to their resistance against offline credential-stealing and online channel-breaking attacks.

personalized login page from the genuine banking server.

Online channel-breaking attacks, as practiced by a malicious man in the middle, a commercially motivated market-scorer, (www.pcworld.com/news/article/0,aid,118757,00.asp), or a security-motivated content-inspector (www.microdasys.com) or Web washer (www.cyberguard.com), are even more sophisticated. Instead of trying to get the user's credentials, the intruder unnoticeably intercepts messages between the client PC and the banking server by masquerading as the server to the client and vice versa. Although the server is normally authenticated via a public-key certificate when a SSL/TLS session is established, users sometimes naively ignore messages about invalid or untrusted certificates or, even worse, are fooled to trust online-generated fake server certificates from a nested intruder certification authority (CA). As a result, an intruder could hijack the authenticated banking session or silently manipulate transaction data. In contrast to offline credential attacks that work decoupled from an actual user-initiated banking session, online channel-breaking attacks don't necessarily compromise the user's credentials but the *session's* credentials

and therefore typically require the user-initiated banking session to work properly.

An attack taxonomy

Now that we've identified the two main attack vectors—credential stealing and channel breaking—we can build an attack taxonomy (see Figure 3). The first step is to look more closely at what makes Internet banking authentication methods so vulnerable to attack in the first place. Offline credential-stealing attacks are effective only against schemes in which user credentials are valid for a rather long time period (vulnerable to phishing) and stored or entered on a potentially insecure device, such as the user's PC (vulnerable to malicious software). The most prominent example is static passwords that are set once and used repeatedly afterward. Such security is based simply on the assumption that the password is nontrivial and kept secret, which in turn requires a trusted environment in which to use the password. Malicious software attacks such as a virus or a Trojan horse, once installed on a client PC, can easily log all keyboard input and periodically email any data gathered to a predefined address. Phishing attacks are even easier to set up because they require very limited context

information (such as the name of the user's bank). After capturing a static password, an attacker can use the password fraudulently for some time without raising the user's suspicion.

A better solution is the use of one-time passwords. The bank sends the user an ordered list of randomly chosen passwords (sometimes called a *scratch list*), each of which is valid for one authentication only. Stealing a one-time password doesn't make much sense because it can't be reused later, but all unused passwords must be kept secret. Unfortunately, some users store their password lists on their PCs for convenience, effectively breaking the underlying security assumption and exposing the passwords to offline credential-stealing attacks. Malicious software can then steal the list at any point in time, not just during authentication. Phishing is also possible, but it's slightly more difficult if the banking server explicitly specifies which one-time password will be used next. Moreover, the legitimate user might eventually notice the fraudulent use of one or more one-time passwords.

To cross the offline-credential-stealing-attack boundary depicted in Figure 3, an authentication method must thwart attacks by both malicious software and phishers. The former requires that credentials are never pre-exposed to a potentially insecure device such as the user's PC, and the latter is rendered infeasible by limiting the validity of a once-exposed credential to a short time period, effectively generating short-lived credentials on demand. Both requirements are usually fulfilled via small microprocessor-based hardware tokens with a built-in display and a cryptographic key unique to the token. The token then uses this key, together with an additional source of entropy (such as the current time from a synchronized clock on the token or a short-lived random challenge from the bank's server entered via a keypad on the token), to generate short-time passwords that are valid for, say, 60 seconds. Because these tokens are stand-alone devices neither directly nor indirectly exposed to the Internet, the user must manually copy the password from the display and enter it in his or her PC. Challenge-response tokens are considered to be slightly more secure than timer-based ones because the additional source of entropy—the challenge—is short-lived, non-deterministic, and (ideally) bound to a previously communicated account number.

Authentication based on a hardware-token public-key infrastructure (PKI) also avoids the risk of offline credential-stealing attacks against insufficiently secured PCs. Specifically, these schemes effectively cross the online-channel-breaking-attack boundary independently of user behavior via a SSL/TLS channel-parameter-dependent challenge. PKI uses asymmetric cryptographic algorithms such as Rivest Shamir Adleman (RSA) or Elliptic Curve Cryptography (ECC). Initially, the bank fits each user with a pair of matching private and public keys for which

some trusted authority issues a matching digital certificate. The certificate attests that the username is associated with the given public key and that the user holds the corresponding private key. The private key and certificate then establish a mutually authenticated SSL/TLS channel between the user's PC and the bank's server, effectively eliminating online channel-breaking attacks. The only critical issue is the protection of the user's private key against malicious software. If the key is stored in a so-called *soft token* (a password-encrypted file on the user's PC), the password and, consequently, the private key are vulnerable to offline credential-stealing attacks. The private key thus must be stored on a tamper-resistant hardware token such as a microprocessor-based smart card, potentially exposing only private-key-related functionality.

Today's smart cards implement a variety of hardware and software countermeasures that can thwart physical as well as logical attacks against the card itself, thus making it almost impossible to steal a private key from a smart card. However, the use of a smart card can still provide a potential point of attack. In most cases, the user must send a private identification number (PIN) to the card to unlock it—only if this happens will the private-key functionality become available. Yet entering and sending a PIN to the smart card via the PC exposes the PIN and thus private-key functionality to malicious software. Despite being very sophisticated and systematic, such attacks are perfectly feasible and can only be eliminated by introducing a certified tamper-resistant smart-card reader. In this way, sensitive operations that require user interaction (such as entering a PIN) move from the potentially exposed PC to a trusted reader device that interacts with the user and the smart card directly through its own secure interfaces.

However, the reader of this article must be aware that to effectively thwart online-channel-breaking-attacks, mutually authenticated SSL/TLS channels must remain stable so that session management can be based on the SSL/TLS session ID. If this isn't the case, and the banking session relies on SSL/TLS channel-independent session credentials (such as cookies) to counter, for example, browser session time-out settings, channel-breaking resistance simply can't be achieved.¹

Let's look more closely at two state-of-the-art Internet banking authentication schemes based on challenge-response: the first one uses short-time passwords and the second uses a secure smart-card reader.

Short-time password solution

Considering the pervasiveness of malicious software and phishing attacks today, any Internet banking solution must be resistant to offline credential-stealing attacks. To achieve this, we propose a challenge/response-based short-time password authentication method that uses symmetric cryptography in combination with a hardware security module (smart card) and an offline (stand-

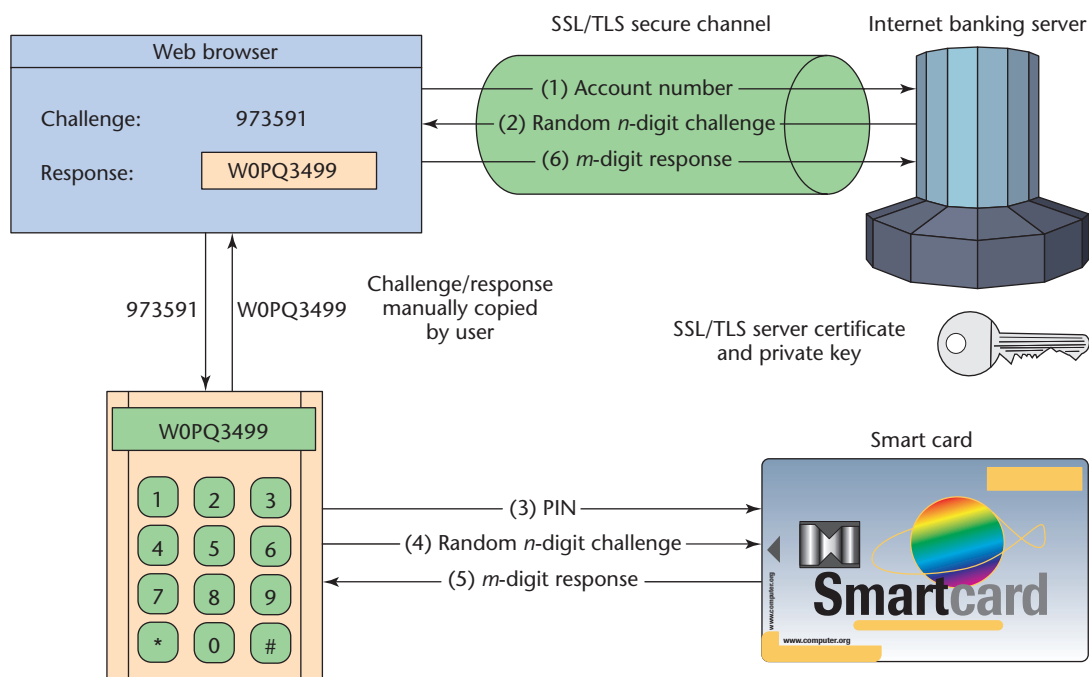


Figure 4. Overview of the short-time password solution. This authentication scheme uses an offline card reader and a smart card to produce short-lived passwords on demand.

alone) smart-card reader (see Figure 4). This solution provides convenient mobility for people who want to do Internet banking anytime anywhere, not just from their homes.

At the core of this scheme is a smart card personalized with a randomly chosen symmetric cryptographic key (such as a DES or AES [Data/Advanced Encryption Standard] key) and a strictly monotonic counter; a PIN protects the card. The use of a symmetric scheme here is crucial for enabling the algorithm's output to be shortened to an appropriately user-convenient, yet sufficiently guessing attack-resistant size. The user communicates with the card via an offline smart-card reader equipped with a small display and keypad. The user can enter the PIN and temporarily open the smart card for further processing via the keypad. For communication with the banking server, standard Web browsers provide the user interface; all Web pages comprise standard HTML code only.

User authentication works as follows:

1. The user connects to the Internet banking server via SSL/TLS with server-side authentication; this way, the user is assured connection with a genuine banking server by explicitly validating the server certificate.
2. The user claims his or her identity by entering an account number in the bank's login form; the banking server displays an n -digit challenge and asks for a matching m -digit response.
3. The user opens his or her smart card by entering the

corresponding PIN in the reader before entering the given challenge. The smart card then calculates the matching response by encrypting the challenge and the incremented on-card login counter with its symmetric cryptographic key; finally, it encodes the result as an appropriately presentable response string.

4. The user manually copies the shown response to the bank's login form to be checked by the bank's authentication server, which redoes the same calculation independently. Because the login counters on the smart card and the server can diverge (if a user playfully calculates a response, for example), the server tries to securely synchronize its local counter within a small range of, say, 32 counter values.

The user's credentials are stored on the tamper-resistant smart card and can only be accessed via an offline smart-card reader, so malicious software can't get the user's symmetric cryptographic key or related functionality. Phishing attacks also won't work because an attacker won't know which challenge the banking server will give next and because the challenges are short-lived and bound to an account number. However, without additional measures, the scheme isn't suitable for crossing the online-channel-breaking-attack boundary independent of user behavior. But given the higher level of sophistication required to launch online channel-breaking attacks, the remaining risk might be acceptable in view of the

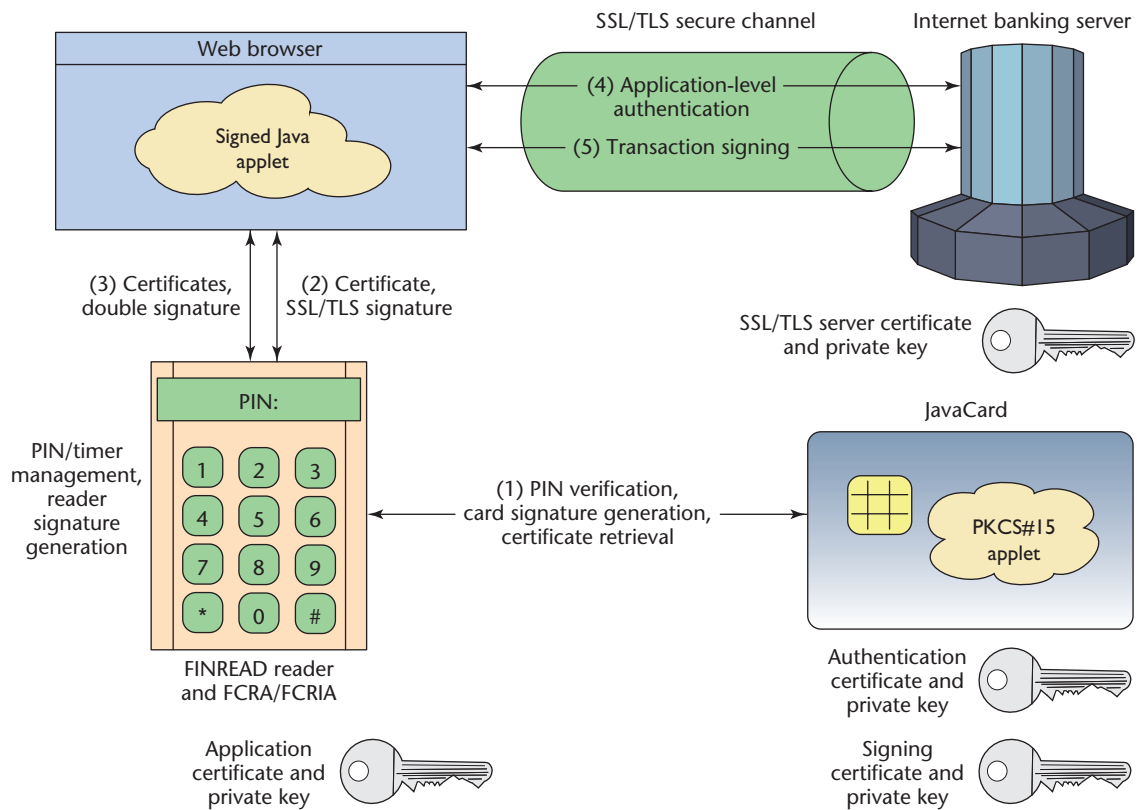


Figure 5. Overview of the certificate-based solution. This authentication scheme uses a secure online card reader and a smart card to sign SSL/TLS challenges on demand.

benefits gained, especially mobility.

Technically, the most elegant way to cross the online-channel-breaking-attack boundary with such a scheme would be an extension to SSL/TLS that supports symmetric as well as asymmetric SSL/TLS client authentication. Instead of today's default SSL/TLS procedure, in which a protocol-data-dependent challenge can only be signed with a user's private key, the improved procedure would require a challenge-response round trip with explicit user interaction and a short-time password authentication scheme. Such a change would be rather nonintrusive: we'd only have to extend the SSL/TLS certificate request to allow the server to ask for an appropriately derived part of the challenge to be displayed on the user's screen and an appropriate response or short-time password to be gathered from the user's keyboard via a personalized window. The challenge-response length and format are thus four independent parameters (for example, a six-digit numeric character challenge and an eight-digit alphanumeric character response). By signing a SSL/TLS channel-parameter-dependent challenge, the authentication response becomes channel-specific—it's valid only for the server for which the SSL/TLS channel has been established, meaning it perfectly resists online

channel-breaking attacks.

Certificate-based solution

In a more stationary setup, crossing the online-channel-breaking-attack boundary independent of user behavior becomes mandatory.² For this, we propose a two-stage, PKI-based Internet banking authentication solution characterized by open standards and a programmable, certified, secure, smart-card reader connected to a potentially exposed PC (see Figure 5).

Similar to the challenge-response short-time password solution, the user receives a smart card that acts as the secure token for his or her Internet banking account. In this case, the smart card includes an advanced microprocessor chip that supports RSA public-key cryptography. Furthermore, it holds a smart-card operating system compliant with the JavaCard specification publicly available from Sun Microsystems ([http://java.sun.com/products.javacard](http://java.sun.com/products/javacard)). JavaCard has become the de facto standard in smart-card operating systems in recent years because it represents a platform-independent multi-application runtime environment based on a Java virtual machine. Because many different sources manufacture JavaCards, such applications (so-called *applets*) are not only indepen-

Related work

The authentication schemes and attacks introduced in the article represent the standard of knowledge discussed in various publications dealing with user authentication.¹ However, most of them provide just an overview of schemes and corresponding attacks and don't attempt to draw a security landscape by relating them to each other in a sensible way.

Short-time password solutions based on a password-generating hardware token are available from various manufacturers such as RSA Security (www.rsasecurity.com), Actividentity (www.actividentity.com), or VeriSign (www.verisign.com). RSA's SecurID solution² is the most prominent example. It consists of a small device (with an LCD display and an internal timer) that continuously calculates the next short-time password. In contrast to the solution presented in the main article, SecurID is timer-based and doesn't use a smart card to ensure tamper resistance and scalable vendor-independent device personalization. Furthermore, because it generally isn't equipped with an alphanumeric or numeric keypad, the short-time password-generation functionality can neither be private identification number (PIN) protected nor can it be extended to transaction signing. Challenge-response-based solutions such as the one described in the main article are available, too, but they're rarely used on a large scale. Clearly, convenience comes before security at this point.

Only a few banks have decided to use public-key cryptography for their Internet banking systems, mostly to avoid setting up and maintaining a public-key infrastructure (PKI). One example in which

a PKI solution is in production is Migrosbank (www.migrosbank.ch/de/Private/KartenZahlungsverkehr/MCardMCardSmart.htm). Here, the e-banking customer uses a smart card to securely store an RSA private key and sign data in the context of a challenge-response authentication protocol. In contrast to the FINancial Transactional IC Card READER- (FINREAD-) based solution presented in the article, this type of PKI solution relies mostly on simple card-reader devices; it's not equipped with a secure keypad, display, or cryptographic capabilities. Advanced solutions that use FINREAD are slowly emerging. Within the EU-funded Trusted FINREAD project,³ a remote banking pilot recently demonstrated the FINREAD platform's basic functionality and interoperability. However, most of today's solutions use keypad-equipped readers; solutions compliant with the home banking computing interface (HBCI; www.hbci-zka.de/english/index.html) use the reader mainly to implement secure PIN entry. With both SSL/TLS client authentication and double signatures authenticating the card as well as the reader device, the solution we've presented in the main text is vastly superior.

References

1. R.E. Smith, *Authentication: From Passwords to Public Keys*, Addison-Wesley, 2001; www.smat.us/crypto/auth/.
2. "RSA SecureID Token," RSA Security, June 2004; www.rsasecurity.com/node.asp?id=1157.
3. "FINREAD Specification," ISO/IEC JTC1/SC17 work item 1.17.34, FINREAD Consortium, 2003; www.finread.com.

dent from the hardware platform itself, but also from the card manufacturer.

In a PKI environment, an applet must be loaded onto the smart card that stores key pairs (and matching certificates) and generates digital signatures. The PKCS#15 Cryptographic Token Information Format Standard,³ defined and maintained by RSA Laboratories, specifies how information such as keys, certificates, or PINs must be structured on a smart card. In this way, we can achieve another level of independence between the on-card application and the PC software communicating with the smart card. The user's smart card—or, more precisely, the PKCS#15 application on the card—is then personalized with an RSA key pair along with a matching certificate issued by a CA the bank controls. The PKCS#15 application protects the private-key functionality with a PIN such that signatures can only be generated if the valid PIN is presented beforehand. Furthermore, the user must be in possession of a FINREAD (FINancial Transactional IC Card READER) smart-card reader connected to his or her PC.

Similar to how JavaCard operates in the world of smart cards, FINREAD is a set of open technical specifications that define the properties of a secure smart-card

reader device, also called a FINREAD card reader (FCR).⁴ An FCR must have certain physical properties such as tamper resistance, a secure display, and a secure keypad. Tamper resistance, in particular, applies to the reader's credentials because an FCR holds its own set of cryptographic keys and can even perform RSA calculations. Like JavaCards, an FCR's firmware also provides a multi-application runtime environment based on a Java virtual machine. Moreover, each FCR can host platform-independent Java applications—FINREAD card reader applications (FCRAs)—that the bank or its partners can load in a strictly controlled way. Most of the keys that the reader hosts are required for reader management operations such as loading FCRAs, maintaining reader firmware, and exchanging the keys themselves. One RSA key, however, is available to FCRAs for application-level purposes.

For our certificate-based solution, we use a dedicated FCR along with an appropriate FINREAD card reader identification application (FCRIA), both of which are loaded onto the user's FCR to secure the authentication process. The FCRIA doesn't allow transparent smart-card access—that is, it's impossible to communicate directly with the smart card from a PC application. The

FCRIA strictly controls all requests that target the card and blocks all security-critical operations such as PIN verification or signature generation. The FCRA is solely responsible for processing these actions, which require explicit approval on the FCR. In addition to a standard programming interface for FCRA and FCRIAs, FINREAD also defines an open-standard client API for interfacing PC applications to FCRA and FCRIAs in a manufacturer-independent way.^{4,5}

The user interface for the Internet banking system is, again, a standard Web browser, and Web pages comprise standard HTML and JavaScript code. However, communicating with the card reader from within a Web browser requires a mechanism to access native code such as a pre-installed DLL. Here, we embed a signed Java applet into the Web page that provides this mechanism via the Java Native Interface (JNI), thus keeping our solution browser-independent (yet not platform-independent). To make the user's certificates visible for the browser's SSL/TLS implementation, two more software components must be installed on the client PC. Netscape Navigator and Mozilla provide a secure token interface in accordance with the PKCS#11 open standard from RSA Laboratories;⁶ the bank can thus use such a PKCS#11 library to connect Netscape Navigator or Mozilla with the FCR or FCRA. Microsoft Internet Explorer, in contrast, uses the Microsoft Crypto API to access keys and certificates, which require a cryptographic service provider to be installed as an interface between Microsoft Internet Explorer and the PKCS#11 library.

User authentication works as follows. First, the user establishes an SSL/TLS channel between the user PC and the bank's Web server by setting up an SSL/TLS session without client authentication. A Java applet running in the Web browser then checks if a FINREAD reader is available and a valid smart card is present in the reader's smart-card slot (if not, the user is requested to insert a valid smart card). Once the card is available, its certificates become visible in the Web browser, and the server initiates an SSL/TLS renegotiation, this time with client authentication. During SSL/TLS client authentication, the generation of a digital signature response on the protocol-data-dependent challenge is required at the client side. The FCRA on the card reader detects this and requests the user to input his or her PIN. If the PIN is valid, the FCRA initiates a signature generation with the authentication key on the card to complete the SSL/TLS client authentication. This establishes an encrypted and mutually authenticated SSL/TLS session over which all communication traffic will be sent. The SSL/TLS channel-parameter-dependent client authentication excludes online channel-breaking attacks, as outlined in the previous section.

Then the bank server initiates an additional user authentication at the application layer. The client receives a

random challenge and forwards it to the FCRA for signature generation. The FCRA reuses the card's authentication key to sign the challenge and then double-signs the signature from the card with the reader's application key. Both signatures are submitted together with the corresponding card and reader certificates to the server for verification. If both signatures are valid and the same client key has been used for both SSL/TLS and application-layer authentication, the banking server is sure that a genuine card is being used in a genuine reader, which eventually excludes offline credential-stealing attacks. To prevent the card from remaining unlocked once the user presents and successfully verifies the PIN, the FCRA maintains timers and counters to appropriately limit the availability of the card's keys.

This scheme effectively thwarts both offline credential-stealing attacks as well as online channel-breaking attacks. Because the FINREAD reader intercepts all calls to the smart card, malicious software can't silently access the smart card and misuse someone else's credentials. Moreover, the SSL/TLS channel-parameter-dependent client authentication eliminates both phishing and online channel-breaking attacks.

Transaction-signing option

Once an authenticated channel is established between the user and the bank, the authorized user can work freely with his or her account and perform all kinds of transactions, including transferring funds and trading shares. However, sophisticated attacks—specifically, manipulating transaction data on the client PC—are still theoretically possible. To thwart such content-manipulation attacks, both of our challenge-based solutions, in contrast to timer-based solutions, provide the option to sign alphanumeric transaction data before submitting it to the server.

In the certificate-based solution, the banking server sends transaction data to the FCRA, and the critical details appear on the FCR's secure display. Given that the user explicitly approves the transaction via the secure keypad, the FCRA double-signs the transaction with the card's signature key (typically, a second RSA private key personalized in the PKCS#15 applet; see Figure 5) and the reader's application key, similar to the application-level authentication step in which a random challenge is signed. By executing critical operations on the trusted reader device and by involving the user via the trusted reader interfaces, content-manipulation attacks can be eliminated. Furthermore, this method allows for tracing and verifying individual transactions and thus also provides means for nonrepudiation.

In the short-time password solution, a similar (although less convenient) protection can be achieved by having the user generate a dedicated password (message authentication code) in response to a user-intelligible

transaction-related challenge such as a beneficiary account number that the user might check. This represents an effective protection against content-manipulation attacks, but because the underlying symmetric key is shared with the bank, it doesn't represent an appropriate means for nonrepudiation, if needed.

Internet banking has turned into an arms race between financial institutions and public network attackers, but with the authentication schemes presented in this article, banks can potentially take a clear lead. Both solutions offer high security against common attacks. From a user perspective, the main difference between the two is that the one-time password scheme supports mobility whereas the certificate-based one is more convenient. With the use of JavaCard, however, it's possible to integrate both solutions on one smart card, thus providing the best of both worlds.

UBS Wealth Management and Business Banking, Zurich, has used the short-time password solution since 2002 and currently counts more than 500,000 users. Despite its inevitable impact on user convenience, the majority of UBS e-banking customers right from the beginning perceived it as a significant security enhancement, compared to the scratch-list-based one-time password solution they were using before. As customers have become even more familiar with the solution, a recent internal market analysis reported an extremely high level of general customer satisfaction with the security but, more important, with overall efficiency and convenience as well. Europay MasterCard Visa (EMV) recently standardized a similar approach with EMV-CAP, its chip authentication program for secure credit-card payments over the Internet.⁷

UBS Wealth Management and Business Banking also piloted the certificate-based solution in 2004, with roughly 100 internal users: the missing readiness of the reader technology and the client PC software platform led to the decision of postponing the spread of this solution to customers. However, changing legislation and the eventually spread of e-IDs among customers makes this solution a highly attractive and valuable alternative for the future.^{8,9} □

References

1. K. Fu et al., "Dos and Don'ts of Client Authentication on the Web," *Proc. 2001 Usenix Security Forum*, Usenix Assoc., 2001, pp. 251–268; www.usenix.org/events/sec01/fu/fu.html.
2. B. Schneier, "Two-Factor Authentication: Too Little, Too Late," *Comm. ACM*, vol. 48, no. 4, Apr. 2005, p. 136.
3. "PKCS#15: Cryptographic Token Information Format Standard," RSA Security, 6 June 2000; [ftp://ftp.rsa-security.com/pub/pkcs/pkcs-15/pkcs-15v1_1.pdf](http://ftp.rsa-security.com/pub/pkcs/pkcs-15/pkcs-15v1_1.pdf).
4. "FINREAD Specification," ISO/IEC JTC1/SC17 work item 1.17.34, FINREAD Consortium, 2003; www.finread.com.
5. "PC/SC Specification Version 2.0," PC/SC Workgroup, 2005; www.pcscworkgroup.com/specifications/specdownload.php.
6. "PKCS#11: Cryptographic Token Interface Standard," RSA Security, June 2004; [ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20.pdf](http://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20.pdf).
7. "Chip Authentication Program—Functional Architecture," MasterCard Int'l, Sept. 2004; available upon request from chip_help@mastercard.com.
8. *Secure User Authentication over a Communication Network*, EU patent 1349032, to UBS AG, European Patent Office, 2003 (US patent pending).
9. *Secure User and Data Authentication over a Communication Network*, EU patent 1349031, to UBS AG, European Patent Office, 2003 (US patent pending).

Alain Hiltgen is a senior information security consultant at UBS Wealth Management and Business Banking, Zurich. His research interests focus on authentication methods, security protocols, key-management processes, and cryptographic techniques. Hiltgen has an MS in electrical engineering and a PhD in technical sciences (cryptology) from the Swiss Federal Institute of Technology (ETH Zurich). He is also a working block member of the European Payments Council's Security of Payments Task Force. Contact him at alain.hiltgen@ubs.com.

Thorsten Kramp is a research staff member in the business computing group at IBM's Zurich Research Laboratory, Switzerland. His research interests include secure systems, smart cards, networking middleware, and business process management. Kramp has a PhD in computer science from the University of Kaiserslautern, Germany. Contact him at thk@zurich.ibm.com.

Thomas Weigold is a software engineer and member of the business computing research group at IBM's Research Laboratory in Zurich, Switzerland. His research interests include secure systems, smart cards, and business process management. He has a diploma in computer science from the Polytechnic University of Konstanz, Germany, and an MSc in computer science from the University of Westminster, UK. Contact him at twe@zurich.ibm.com.