Using a Personal Device to Strengthen Password Authentication from an Untrusted Computer*

Mohammad Mannan and P. C. van Oorschot

School of Computer Science
Carleton University, Ottawa, Canada

Abstract. Keylogging and phishing attacks can extract user identity and sensitive account information for unauthorized access to users' financial accounts. Most existing or proposed solutions are vulnerable to session hijacking attacks. We propose a simple approach to counter these attacks, which cryptographically separates a user's long-term secret input from (typically untrusted) client PCs; a client PC performs most computations but has access only to temporary secrets. The user's long-term secret (typically short and low-entropy) is input through an independent personal trusted device such as a cellphone. The personal device provides a user's long-term secrets to a client PC only after encrypting the secrets using a pre-installed, "correct" public key of a remote service (the intended recipient of the secrets). The proposed protocol (`MP-Auth`) realizes such an approach, and is intended to safeguard passwords from keyloggers, other malware (including rootkits), phishing attacks and pharming, as well as to provide transaction security to foil session hijacking. We report on a prototype implementation of MP-Auth, and provide a comparison of web authentication techniques that use an additional factor of authentication (e.g. a cellphone, PDA or hardware token).

1 Introduction

Passwords enjoy ubiquitous use for online authentication. Although many more secure (typically also more complex and costly) authentication protocols have been proposed, the use of passwords for Internet user authentication remains predominant. Due to the usability and ease of deployment, most financial transactions over the Internet are authenticated through a password. Hence passwords are a prime target of attackers, for economically-motivated exploits including those targeting online bank accounts and identity theft.

Online banking – as one example of highly critical Internet services – often requires only a bank card number (as *userid*) and password. Users input these credentials to a bank website to access their accounts. An attacker can easily collect these long-term secrets by installing a keylogger program on a client PC, or embedding a JavaScript keylogger [29] on a phishing website. In today's Internet environment, software keyloggers are typically installed on a user PC

* Version: March 30, 2007. Contact author: `mmannan@scs.carleton.ca`.

along with common malware and spyware [25]. An increasing number of phishing sites also install keyloggers on user PCs, even when users do not download or click any link on those sites [1]. Client security is a big problem, regardless of the software/hardware platform used, as when plaintext sensitive information is input to a client PC, such malware has instant access, compromising (reusable) long-term secrets. We argue that for some common applications, passwords are too important to input directly to a typical user PC on today's Internet; and that the user PC should no longer be trusted with such plaintext long-term secrets, which are intended to be used for user authentication to a remote server.

To safeguard a long-term password, we build on the following simple idea: use a hand-held personal device, e.g., a cellphone or PDA to encrypt the password (combined with a server generated random challenge) under the public key of an intended server, and relay through a (possibly untrusted) PC only the encrypted result in order to login to the server website. This simple challenge-response effectively turns a user's long-term password into a one-time password in such a way that long-term passwords are not revealed to phishing websites, or keyloggers on the untrusted PC.

The resulting protocol, called `MP-Auth` (short for *M*obile *P*assword *Auth*entication), is proposed primarily to protect a user's long-term password input through an untrusted (or rather, untrustworthy) client PC. For usability and other reasons, the client PC is used for the resulting interaction with the website, and performs most computations (e.g. session encryption, HTML rendering etc.) but has access only to temporary secrets. The capabilities we require from a mobile device include encryption, alpha-numeric keypad, short-range network connection (wire-line or Bluetooth), and a small display. Although we highlight the use of a cellphone, the protocol can be implemented using any similar "trustworthy" device (e.g. PDAs or smart phones), i.e., one free of malware. There are known attacks against mobile devices [11], but the trustworthiness of such devices is currently more easily maintained than a PC, in part because they contain far less software; see Section 3.2 for further discussion of mobile device security. The use of a mobile device in MP-Auth is intended to protect user passwords from easily being recorded and forwarded to malicious parties, e.g., by keyloggers installed on untrustworthy commodity PCs.

Another simple attack to collect user passwords is phishing. Although phishing attacks have been known for at least 10 years (see [10]), few, if any, anti-phishing solutions exist today that are complete and deployable. In MP-Auth, we encrypt a password with the "correct" public key of a web server (e.g. a bank), so that the password is not revealed to any phishing websites. MP-Auth is intended to protect passwords from keyloggers as well as various forms of phishing (including deceptive malware, DNS-based attacks or *pharming*, as well as false bookmarks). New malware attacks (*bank-stealing Trojans* or *session-hijacking*, e.g. Win32.Grams [5]; see also CERT [22]) attempt to perform fraudulent transactions in real-time after a user has logged in, instead of collecting userids and passwords for later use. Most existing or proposed solution techniques are susceptible to these new attacks, e.g., Phoolproof [27] (presented in FC'06), and two-factor authentication such as a password and a passcode generator token (e.g. SecurID). MP-Auth protects against session hijacking, by providing transaction integrity through a transaction confirmation step. Unlike standard two-factor techniques, MP-Auth does not store any secret on the mobile device.

Much of the related work in the literature concerns the trustworthiness of *public* computers, e.g., in Internet cafés and airport lounges. Home computers are generally assumed to be trusted. Solutions are primarily designed to deal with the problem of untrusted computers in public settings. In reality, most user PCs are not safe anywhere; an improperly patched computer – home or public – generally survives only minutes[1] when connected to the Internet. There are also now many anti-phishing proposals (e.g. [29], [36]), and software "tools" designed to detect spoofed websites (e.g. eBay toolbar, SpoofGuard, Spoofstick, Netcraft toolbar). However, most of these are susceptible to keylogging attacks.[2] On the other hand, several authentication schemes which use a trusted personal device, generally prevent keyloggers, but do not help against phishing or session hijacking attacks. In contrast, the goal of MP-Auth is to protect passwords from both keyloggers and phishing sites, and provide transaction security.

Our Contributions. We propose MP-Auth, a protocol for online authentication using a personal device such as a cellphone in conjunction with a PC. The protocol provides the following benefits without requiring a trusted proxy (e.g. [7]), or storing a long-term secret on a cellphone (e.g. Phoolproof [27]).

1. Keylogging Protection. A client PC does not have access to long-term user secrets. Consequently keyloggers (software or hardware) on the PC cannot access critical passwords.
2. Phishing Protection. Even if a user is directed to a spoofed website, the website will be unable to decrypt a user password. Highly targeted phishing attacks (*spear phishing*) are also ineffective against MP-Auth.
3. Pharming Protection. In the unlikely event of domain name hijacking [14], MP-Auth does not reveal a user's long-term password to attackers. It also protects passwords when the DNS cache of a client PC is poisoned.
4. Transaction Integrity. With the transaction confirmation step (see Section 2) in MP-Auth, a user can detect any unauthorized transaction during a login session, even when an attacker has complete control over the user PC (through e.g. SubVirt [16] or Blue Pill [30]).
5. Applicability to ATMs. MP-Auth is suitable for use in ATMs, if an interface is provided to connect a cellphone, e.g., a wire-line or Bluetooth interface. This can be a step towards ending several types of ATM fraud (see Bond [4] for a list of ATM fraud cases).

We analyzed MP-Auth using AVISPA [2]; no attacks were found. We have also implemented a prototype of MP-Auth for performance testing.

[1] The average time between attacks is reported to be 5 minutes as of March 26, 2007; see at `http://isc.sans.org/survivaltime.html`.

[2] PwdHash [29] can protect passwords from JavaScript keyloggers, but not software keyloggers on client PCs.

Organization. The MP-Auth protocol, threat model and operational assumptions are discussed in Section 2. A brief analysis of MP-Auth messages, and circumstances under which MP-Auth fails to provide protection are outlined in Section 3. Discussion on usability and deployment issues related to MP-Auth are provided in Section 4. Section 5 summarizes our MP-Auth prototype implementation. Related work is discussed in Section 6. Section 7 concludes.

2 MP-Auth: A Protocol for Online Authentication

We now describe the MP-Auth protocol, including threat model assumptions.

Threat Model and Operational Assumptions. The primary goals of MP-Auth are to protect user passwords from malware and phishing websites, and to provide transaction integrity. We assume that a bank's "correct" public key is available to users (see below for discussion on public key installation). We assume that mobile devices are malware-free. A browser on a PC uses a bank's SSL certificate to establish an SSL connection with the bank website (as per common current practice). The browser may be duped to go to a spoofed website, or have a wrong SSL certificate of the bank or the verifying certificating authority. The protocol does not protect user privacy (of other than the user's password) from an untrusted PC; the PC can record all transactions, generate custom user profiles etc. Visual information displayed to a user on a PC screen is also not authenticated by MP-Auth, i.e., a malicious PC can display misleading information to a user without being (instantly) detected. Denial-of-service (DoS) attacks are not addressed.

Protocol Steps in MP-Auth. For notation see Table 1. Before the protocol begins, we assume that user U's cellphone M is connected to B (via wire-line or Bluetooth). The protocol steps are described below (see also Fig. 1).

U, M, B, S	User, a cellphone, a browser on the user PC, and the server, respectively.
ID_S, ID_U	Server ID and user ID, respectively. ID_U is unique in the server domain.
P	Long-term (pre-established) password shared between U and S.
R_S	Random number generated by S.
$\{data\}_K$	Symmetric (secret-key) encryption of *data* using key K.
$\{data\}_{E_S}$	Asymmetric (public-key) encryption of *data* using S's public key E_S.
X, Y	Concatenation of X and Y.
K_{BS}	Symmetric encryption key shared between B and S (e.g. an SSL key).
$f(\cdot)$	A cryptographically secure hash function.
$v(\cdot)$	A visualization function that maps any arbitrary binary string into easy-to-read words [12].

Table 1. Notation used in MP-Auth

1. U launches a browser B on the untrusted PC, and visits the bank website S.
2. B and S establish an SSL session; let K_{BS} be the established SSL secret key.
3. S generates a random nonce R_S, and sends the following message to B.

$$B \leftarrow S : \ \{ID_S, R_S\}_{K_{BS}} \tag{2.1}$$

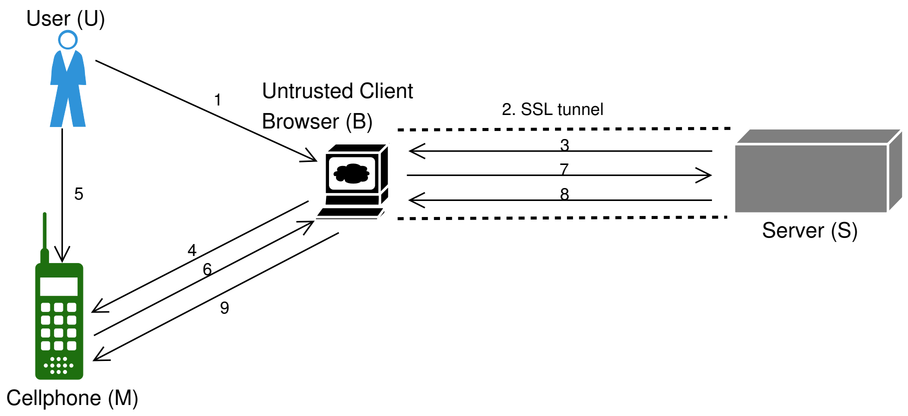

Fig. 1. MP-Auth protocol steps

4. B decrypts message (2.1) and forwards it to M.

$$M \leftarrow B: \; ID_S, R_S \tag{2.2}$$

   We describe an additional step called *session ID verification* (see below) in cases where protecting the integrity of R_S is useful.
5. M prompts the user to input the userid and password for S. A userid (e.g. bank card number) may be stored on the cellphone for convenience; the password should not be stored or auto-remembered.
6. M generates a random secret nonce R_M and encrypts R_M using E_S. M calculates the session key K_{MS} and sends message (2.4) to B (here, the userid ID_U is, e.g., a bank card number).

$$K_{MS} = f(R_S, R_M) \tag{2.3}$$
$$M \rightarrow B: \; \{R_M\}_{E_S}, \{f(R_S), ID_U, P\}_{K_{MS}} \tag{2.4}$$

7. B (via SSL) encrypts message (2.4) with K_{BS}, and forwards the result to S.
8. From message (2.4), after SSL decryption, S decrypts R_M using its corresponding private key, calculates the session key K_{MS} (as in equation (2.3)), decrypts the rest of message (2.4), and verifies P, ID_U and R_S. Upon successful verification, S grants access to B on behalf of U. S sends the following message for M to B (indicating login success).

$$B \leftarrow S: \; \{\{f(R_M)\}_{K_{MS}}\}_{K_{BS}} \tag{2.5}$$

9. B forwards $\{f(R_M)\}_{K_{MS}}$ to M. M decrypts to recover $f(R_M)$ and verifies its local copy of R_M. Then M displays success or failure to U.

Transaction Integrity Confirmation. In MP-Auth, M and S establish a session key K_{MS} known only to them; malware on a user PC has no access to K_{MS}. Attackers may modify or insert transactions through the untrusted PC.

To detect and prevent such transactions, MP-Auth requires explicit transaction confirmation by U (through M). The following messages are exchanged (after step 9) for confirmation of a transaction with summary details T (R_{S1} is a server generated random nonce, used to prevent replay).

$$M \xleftarrow{\{T, R_{S1}\}_{K_{MS}}} B \xleftarrow{\{\{T, R_{S1}\}_{K_{MS}}\}_{K_{BS}}} S \qquad (2.6)$$

$$M \xrightarrow{\{f(T, R_{S1})\}_{K_{MS}}} B \xrightarrow{\{\{f(T, R_{S1})\}_{K_{MS}}\}_{K_{BS}}} S \qquad (2.7)$$

M displays T to U in a human-readable way (e.g. "Pay $10 to Vendor V from the Checking account"), and asks for confirmation (yes/no). When the user confirms T, the confirmation message (2.7) is sent from M to S (via B). From message (2.7), S retrieves $f(T, R_{S1})$, and verifies with its local copy of T and R_{S1}. Upon successful verification, T is committed. Instead of initiating a confirmation step after each transaction, transactions may be confirmed in batches (e.g. four transactions at a time); then, T will represent a batch of transactions in the above message flows.

In an environment where a client machine is less likely to have malware, e.g., an ATM, transaction confirmation may not be needed, if the session ID verification step (see below) is implemented. Also, some transactions may not require confirmation. For example, setting up an online bill payment for a phone company should require user confirmation, but when paying a monthly bill to that account, the confirmation step can be omitted. Fund transfers between user accounts without transaction confirmation may pose no significant risks to users. A bank may configure the set of sensitive transactions that will always require the confirmation step (a user may also add to that set).

Session ID Verification. To detect modification to R_S (when being forwarded to M), we add a session ID verification step after step 4. Both B and M compute a session ID $sid = v(R_S)$. B and M display sid to U. U proceeds only if both session IDs are the same. To minimize user errors, M shows a list of session IDs (one derived from R_S and others chosen randomly), and asks U to select the correct sid corresponding to the one displayed on B.

We assume that users will be able to distinguish differences in sid, especially when sid is easily human-verifiable, e.g., plain English words, distinct images. Note that malware on a PC can display any arbitrary sequence of words or images. Hence the session ID verification step may only help for ATMs (where we assume an attacker may install a false keyboard panel and card reader on an otherwise trustworthy ATM). When a user accesses an online bank website from a PC, the transaction confirmation step must be implemented; omitting session ID verification in such a case may allow attackers (view-only) access to the user account, but the attackers cannot perform any (meaningful) transaction. (Note that for only viewing a user's transactions, attackers can deploy simple malware on the user PC to capture images of web pages containing the transactions.)

Password Setup/Renewal. In order to secure passwords from keyloggers during password renewal, we require that the password is entered through the cellphone keypad. We assume that the initial password is set up via a trustworthy out-of-band method (e.g. regular phone, postal mail), and U attempts a password renewal after successfully logged into S (i.e. K_{MS} has been established between M and S). The following message is forwarded from M to S (via B) during password renewal (P_{old} and P_{new} are the old and new passwords respectively).

$$M \xrightarrow{X,\text{ where } X = \{ID_U, P_{old}, P_{new}\}_{K_{MS}}} B \xrightarrow{\{X\}_{K_{BS}}} S \qquad (2.8)$$

Public Key Installation. One of the greatest practical challenges of deploying public key systems is the distribution and maintenance (e.g. revocation, update) of public keys. MP-Auth requires a service provider's public key to be distributed (and updated when needed) and installed into users' cellphones. The distribution process may vary depending on service providers; we recommend that it not be primarily Internet-based. Considering banking as an example, we visualize the following key installation methods (but note that we have not user-tested these for usability): (i) at a bank branch, preferably during an account setup, (ii) through in-branch ATM interfaces (hopefully free of "fake" ATMs), (iii) through a cellphone service (authenticated download) as data file transfer.

A challenge-response protocol or integrity cross-checks (using a different channel, e.g., see [34]) should be used to verify the public key installed on a cellphone, in addition to the above procedures. For example, the bank may publish its public key on the bank website, and users can cross-check the received public key (e.g. comparing *visual hashes* [28]).

Requirements and Drawbacks of MP-Auth. MP-Auth requires users possess a malware-free (see Section 3.2) personal device. Public keys of each target website (e.g. bank) must be installed on the personal device. (We assume that there are only a few financially critical websites that a typical user deals with.) The correctness, i.e., integrity of installed public keys must also be maintained. A communication channel between a personal device and PC is needed, in such a way that malware on the PC cannot infect the personal device.[3] For ATMs, users must compare easy-to-read words [12] or easily distinguishable images [28] generated from random binary strings.

3 Security and Attack Analysis

In this section, we provide a brief informal security analysis of MP-Auth. We motivate a number of design choices in MP-Auth messages and their security implications. We also list successful but less likely attacks against MP-Auth.

As a confidence building step, we have tested MP-Auth using the AVISPA (Automated Validation of Internet Security Protocols and Applications) [2] analysis tool, and found no attacks. AVISPA is positioned as an industrial-strength technology for the analysis of large-scale Internet security-sensitive protocols and applications. AVISPA test code for MP-Auth is available [17]. We have not at this point carried out other formal analyses or security proofs for MP-Auth.

[3] The first *crossover* virus was reported [23] in February 2006.

3.1 Partial Message Analysis and Motivation

Here we provide motivation for various protocol messages and message parts. In message (2.1), S sends a fresh R_S to B, and B forwards ID_S, R_S to M. ID_S is included in message (2.2) so that M can choose the corresponding public key E_S. When U starts a session with S, a nearby attacker may start a parallel session from a different PC, and grab M's response message (2.4) (off-the-air, from the Bluetooth connection) to login as U. However, as S generates a new R_S for each login session (i.e. U and the attacker receive different R_S from S), sending message (2.4) to S by any entity other than B would cause a login failure.

The session key K_{MS} shared between M and S, is known only to them. Both M and S influence the value of K_{MS} (see equation (2.3)), and thus a sufficiently random K_{MS} is expected if either of the parties is honest (as well as capable of generating secure random numbers); i.e., if a malicious party modifies R_S to be 0 (or other values), K_{MS} will still be essentially a random key when M chooses R_M randomly. To retrieve P from message (2.4), an attacker apparently must guess K_{MS} (i.e. R_M) or S's private key. If both these quantities are sufficiently large (e.g. 160-bit R_M and 1024-bit RSA key E_S) and random, an offline dictionary attack on P becomes computationally infeasible. We encrypt only a small random quantity (e.g. 160-bit) by E_S, which should always fit into one block of a public key cryptosystem (including elliptic curve). Thus MP-Auth requires only one public key encryption. Browser B does not have access to K_{MS} although B helps M and S establish this key. With the transaction integrity confirmation step, all (important) transactions must be confirmed from M using K_{MS}; therefore, any unauthorized (or modified) transaction by attackers will fail as attackers do not have access to K_{MS}.

3.2 Attacks Against MP-Auth

Although MP-Auth apparently protects user passwords from malware installed on a PC or phishing websites, here we discuss some other possible attacks against MP-Auth which, if successful, may expose a user's plaintext password.

a) Mobile Malware. We have stated the requirement that the personal (mobile) device be trusted. An attack could be launched if attackers can compromise mobile devices, e.g., by installing a (secret) keylogger. Malware in mobile networks is increasing as high-end cellphones (smart phones) contain millions of lines of code. For example, a Sept. 2006 study [11] reported that the number of existing malware for mobile devices is nearly 162 (in comparison, there are more than $100,000$ viruses in the PC world[4]). Worms such as Cabir [8] are designed to spread in smart phones by exploiting vulnerabilities in embedded operating systems. Regular cellphones which are capable of running J2ME MIDlets have also been targeted, e.g., by the RedBrowser Trojan [9]. However, currently cellphones remain far more trustworthy than PCs, thus motivating our proposal.

In the future, as mobile devices increasingly contain much more software, the requirement of trustworthy cellphones becomes more problematic, and their use for sensitive purposes such as online banking makes them a more attractive

[4] `http://www.cknow.com/vtutor/NumberofViruses.html`

target. Limited functionality devices (with less software, implying more trustworthy) may then provide an option for use with MP-Auth. Even if MP-Auth is implemented in such a special-purpose (or lower functionality) device, the device can hold several public keys for different services; in contrast, users may require a separate passcode generator for each service they want to access securely in standard two-factor authentication proposals. Another possibility of restricting mobile malware may be the use of micro-kernels [13], formally verifiable OS kernels [33], protections against virtual-machine based rootkits (VMBRs) [16], or a virtualized Trusted Platform Module (vTPM) [31] on cellphones to restore a trustworthy application environment. The Trusted Computing Group's (TCG's) Mobile Phone Work Group (MPWG) is currently developing specifications [24] for securing mobile phones. Anti-virus software (e.g. Trend Micro [32]) for mobile platforms may also help maintain trustworthiness of cellphones. Malware targeting mobile phones is still limited, and leveraging the experience of working to secure traditional PC platforms may help us achieve a relatively secure mobile computing environment. However, considering the current state of mobile phone security, MP-Auth would perform better on devices whose software upgrade is tightly controlled (e.g. only allowing applications which are digitally signed by a trustworthy vendor).

b) Common-Password Attacks. Users often use the same password for different websites. To exploit such behavior, in a common-password attack, attackers may break into a low-security website to retrieve userid/password pairs, and then try those in financially critical websites, e.g., for online banking. MP-Auth itself does not address the common-password problem (but see e.g., PwdHash [29]).

c) Social Engineering. Some forms of social engineering remain a challenge to MP-Auth (and apparently, other authentication schemes using a mobile device). For example, malware might prompt a user to enter the password directly into an untrusted PC, even though MP-Auth requires users to enter passwords only into a cellphone. In a "mixed" phishing attack,[5] emails are sent instructing users to call a phone number which delivers, by automated voice response, a message that mimics the target bank's own system, and asks callers for account number and PIN. User habit or user instruction may provide limited protection against these.

4 Usability and Deployment

In this section, we discuss usability and deployment issues related to MP-Auth. Usability is a great concern for any protocol supposed to be used by general users, e.g., for Internet banking and ATM transactions. In MP-Auth, users must connect a cellphone to a client PC. This step is more user-friendly when the connection is wireless, e.g., Bluetooth, than wire-line. Then the user browses to a bank website, and enters into the cellphone the userid and password for the site (step 5 in MP-Auth, see Section 2). In ATMs, the password is entered if session ID verification is successful. We also assume that typing a userid and password on a cellphone keypad is acceptable in terms of usability, as many users are

[5] `http://www.cloudmark.com/press/releases/?release=2006-04-25-2`

accustomed to type SMS messages or have been trained by BlackBerry/Treo experience. However, verification of session ID and transactions may be challenging to some users. We have not conducted any user study to this end.

During authentication the cryptographic operations a cellphone is required to perform in MP-Auth include: one public key encryption, one symmetric encryption and one decryption, one random number generation, and three cryptographic hash operations. The most expensive is the one public key encryption, which is a relatively cheap RSA encryption with short public exponent in our application; see Section 5 for concrete results. We now discuss other usability and deployment aspects which may favor MP-Auth (see also Section 6).

1. As it appears from the current trend in online banking (see [18]), users are increasingly required to use two-factor authentication (e.g. with a separate device such as a SecurID passcode generator) for login. Hence using an existing mobile device for online banking relieves users from carrying an extra device. Also, a user might otherwise require multiple hardware tokens for accessing different online accounts (from different banks).
2. The usability of four login techniques has been studied by Wu et al. [35] – two that send a one-time password as an SMS message, visually checking the session names displayed on the phone and untrusted PC, and choosing the correct session ID from a list of choices on the cellphone. Typing a one-time password is least preferred, yet in most two-factor authentication methods in practice, users must do so. In contrast, MP-Auth requires users to enter only long-term passwords. MP-Auth may also require users to compare session IDs by choosing from a list, which is reported to be more secure (the least spoofable of all) and easier than typing a one-time password [35].
3. MP-Auth offers cost efficiency for banks – avoiding the cost of providing users with hardware tokens (as well as the token maintenance cost). The software modification at the server-end is relatively minor; available SSL infrastructure is used with only three extra messages (between a browser and server) beyond SSL. MP-Auth is also compatible with the common SSL setup, i.e., a server and a client authenticate each other using a third-party-signed certificate and a user password respectively.
4. Several authentication schemes involving a mobile device store long-term secrets on the device. Losing such a device may pose substantial risk to users. In contrast, losing a user's cellphone is inconsequential to MP-Auth assuming no *secret* (e.g. no "remembered password") is stored on the phone.
5. Public key distribution and renewal challenges usability in any PKI. Key updating is also troublesome for banks. However, key renewal is an infrequent event; we assume that users and banks can cope with this process once every two to three years. If key updates are performed through the mobile network or selected ATMs (e.g. within branch premises), the burden of key renewal is largely distributed. For comparison, hardware tokens (e.g. SecurID) must be replaced approximately every two to five years.

The above suggests that compared to available two-factor authentication methods, MP-Auth may be as usable or better. However, we hesitate to make strong statements without usability tests (c.f. [6]).

5 Implementation and Performance

We developed a prototype of the main authentication and session key establishment parts of MP-Auth to evaluate its performance. Our prototype consists of a web server, a Firefox Extension, a desktop client, and a MIDlet on the cellphone. We set up a test web server (bank), and used PHP `OpenSSL` functions and `mcrypt` module for the server-side cryptographic operations. The Firefox Extension communicates between the web server and desktop client. The desktop client forwards messages to and from the cellphone over Bluetooth. We did not have to modify the web server or Firefox browser for MP-Auth besides adding PHP scripts to the login page (note that Phoolproof [27] requires browser modifications). We used the `BlueZ` Bluetooth protocol stack for Linux, and Rococosoft's Impronto Developer Kit for Java. We developed a MIDlet – a Java application for Java 2 Micro Edition (J2ME), based on the Mobile Information Device Profile (MIDP) specification – for a Nokia E62 phone. For cryptographic operations on the MIDlet, we used the Bouncy Castle Lightweight Crypto API.

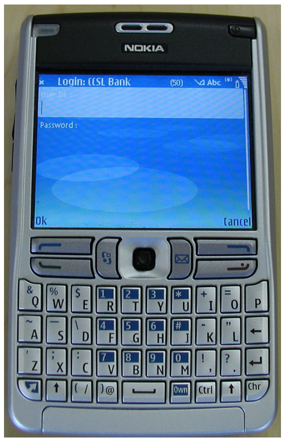

Fig. 2. MP-Auth login

To measure login performance, we used MP-Auth for over 200 successful logins, and recorded the required time (excluding the user input time, i.e. userid and password). We carried out similar tests for regular SSL logins. The results are summarized in Table 2. We use `RSA` 1024-bit and `AES-128` (CBC) for public and symmetric key encryption respectively. `SHA-1` (160-bit) is used as hash function, and `/dev/urandom` and `SecureRandom` (Java) are used as sources of randomness. Although regular SSL login is almost eight times faster than MP-Auth, on average, it takes less than a second for MP-Auth login. We believe that this added delay would be acceptable, given that entering a userid and password takes substantial additional time.

	Avg. Time (s)	[Min, Max] (s)
MP-Auth	0.62	[0.34, 2.28]
Regular SSL	0.08	[0.06, 0.22]

Table 2. Performance comparison between MP-Auth and regular SSL login

6 Related Work

The most common types of existing online authentication techniques include password-only authentication, two-factor authentication, and transaction security mechanisms (e.g. secret SMS to a user's cellphone). Several solutions using a trusted device have been proposed, e.g., Phoolproof [27], BitE [20], camera-based authentication [7]. Due to page limits, our comparison of MP-Auth with related work is limited here; for more extensive discussion, see [18].

In contrast to two-factor authentication methods, by design MP-Auth does not provide attackers any window of opportunity when authentication messages (i.e. collected regular and one-time passwords of a user) can be replayed to login as the legitimate user and perform transactions on the user's behalf. The key observation is that, through a simple challenge-response, message (2.4) in MP-Auth (Section 2) effectively turns a user's long-term static password into a one-time password in such a way that long-term passwords are not revealed to phishing websites, or keyloggers on an untrusted PC. In contrast to transaction security mechanisms, MP-Auth protects both large and small transactions. Also, MP-Auth does not require text or voice communications airtime for web authentication or transaction security. (See also Section 4 for more comparison on usability and deployment issues.)

Balfanz and Felten [3] introduced the *splitting trust* paradigm to split an application between a small trusted device and an untrusted computer. Our work is based on such a paradigm where we provide the long-term password input through widely available cellphones, and use the untrusted computer for computationally intensive processing and display.

Parno et al. [27] proposed Phoolproof, a cellphone-based technique to protect users against phishing with less reliance on users making *secure* decisions. With the help of a pre-shared secret – established using an out-of-band channel, e.g., postal mail – a user sets up an account at the intended service's website. The user's cellphone generates a key pair $\{K_U, K_U^{-1}\}$, and sends the public key to the server. The user's private key and server certificate are stored on a cellphone for logins afterward. During login, a user provides userid and password to a website on a browser (as usual), while in the background, the browser and server authenticate (using SSL mutual authentication) through the pre-established client/server public keys in an SSL session (the browser receives the client public key from the cellphone).

Phoolproof assumes that users can correctly identify websites at which they want to set up an account. Users must *revoke* public/private key pairs in case of lost or malfunctioning cellphones, or a replacement of older cellphone models. Expecting non-technical users (e.g. typical bank customers) to understand concepts of creation and revocation of public keys may not be practical. In MP-Auth, users do not have to revoke any key or inform their banks when they lose, break or change their cellphones.

It is also assumed in Phoolproof that the (Bluetooth) channel between a browser and cellphone is secure. Seeing-is-believing (SiB) [21] techniques are proposed to secure local Bluetooth channels, requiring users to take snapshots using a camera-phone, apparently increasing complexity to users. In MP-Auth, we do not rely on the assumption that the local channel (between the cellphone and PC) is secure. Although MP-Auth may require users to visually verify a session ID to secure the local Bluetooth connection (for ATMs, when transaction integrity confirmation is omitted), users are not required to have a camera-phone or to take any picture. Also, Phoolproof does not aim to protect against session hijacking attacks.

Comparing MP-Auth with Existing Literature. Table 3 summarizes a comparison of MP-Auth with several anti-phishing proposals from the literature. An (✗) means a special requirement is needed. An empty box indicates the stated protection is not provided (first three columns) and the stated requirement is not needed (last four columns). A (—) represents non-applicability. (All ✓and no ✗ would be optimal.) For example, Phoolproof [27] provides protection against phishing and keylogging, but it is vulnerable to session hijacking; it requires a malware-free mobile and stores long-term secrets on the mobile, but does not require a trusted proxy or trusted PC OS. We acknowledge that although this table may provide useful high-level overview, this does not depict an apple-to-apple comparison. Several solutions listed here require a trusted proxy, thus introduce an extra deployment burden, and present an attractive target to determined attackers (providing access to many user accounts). Also, fraudsters may increasingly target mobile devices if long-term secrets are stored on them.

	Protection against			Requirement			
	Session-hijacking	Phishing	Key-logging	Trusted proxy	On-device secret	Trusted PC OS	Malware-free mobile
MP-Auth	✓	✓	✓				✗
Phoolproof [27]		✓	✓		✗		✗
BitE [20]			✓		✗	✗	✗
SpyBlock [15]	✓	✓	✓		—	✗	
Three-party [26]	—	—	✓		✗		✗
Camera-based [7]	✓	✓	✓	✗	✗		✗
Web-Auth [35]		✓	✓	✗	✗		✗
Guardian [19]			✓		✗		✗

Table 3. Comparing MP-Auth with existing literature. For details, see [18].

7 Concluding Remarks

We have proposed MP-Auth, a protocol for web authentication which is resilient to keyloggers (and other malware including rootkits), phishing websites, and session hijacking. Recently, many small-scale, little-known malware instances have been observed that install malicious software launching keylogging and phishing attacks; these are in contrast to large-scale, high-profile worms like Slammer. One reason for this trend might be the fact that attackers are increasingly targeting online financial transactions.[6] Furthermore, such attacks are fairly easy to launch; for example, attackers can gain access to a user's bank account simply by installing (remotely) a keylogger on a user PC and collecting the user's banking access information (userid and password). MP-Auth is designed to prevent such attacks. MP-Auth primarily focuses on online banking but can be used for general web authentication systems as well as at ATMs. Our requirement for a trustworthy personal device (i.e. free of malware) is important, and becomes more challenging over time, but as discussed in Section 3.2, may well remain viable. In our MP-Auth implementation, cryptographic computations and Bluetooth communications took less than a second for login (excluding the user

[6] According to a July 2006 report [1], 93.5% of all phishing sites target online financial services, e.g., online banking and credit card transactions.

input time), which we believe is an acceptable delay. Despite a main objective of preventing phishing and keylogging attacks, MP-Auth as presented remains one-factor authentication; thus an attacker who nonetheless learns a user password can impersonate that user. Consequently, the server side of MP-Auth must be trusted to be secure against both from insider attack and break-in.

Users often input reusable critical identity information to a PC other than userid/password, e.g., a passport number, social security number, driver's licence number, or credit card number. Such identity credentials are short, making them feasible to enter from a cellphone keypad. In addition to protecting a user's userid/password, MP-Auth may easily be extended to protect other identity credentials from the reach of online attackers, and thereby might be of use to reduce online identity theft. We believe that the very simple approach on which MP-Auth is based – using a cellphone or similar device to asymmetrically encrypt passwords and one-time challenges – is of independent interest for use in many other applications, e.g., traditional telephone banking directly from a cellphone, where currently PINs are commonly transmitted in-band without encryption.

We reiterate that although based on a very simple idea, MP-Auth has yet to be user-tested for usability; this (together with [18]) is an architecture and state-of-the-art paper. We encourage the security community to pursue alternate proposals for password-based online authentication which simultaneously address phishing, keylogging and session hijacking rootkits.

Acknowledgements

We thank anonymous reviewers for their constructive comments, Bryan Parno for allowing us access to his Phoolproof [27] implementation, and Masud Khan for providing a Nokia E62 smartphone. The first author is supported in part by an NSERC CGS. The second author is Canada Research Chair in Network and Software Security, and is supported in part by an NSERC Discovery Grant, and the Canada Research Chairs Program.

References

1. Anti-Phishing Working Group. Phishing Activity Trends Report, July, 2006.
2. Armando et al. The AVISPA tool for the automated validation of Internet security protocols and applications. In *Computer Aided Verification (CAV)*, volume 3576 of *LNCS*, 2005. Project website, `http://www.avispa-project.org`.
3. D. Balfanz and E. Felten. Hand-held computers can be better smart cards. In *USENIX Security*, 1999.
4. M. Bond. Phantom withdrawals: On-line resources for victims of ATM fraud. `http://www.phantomwithdrawals.com`.
5. CA Virus Information Center. Win32.Grams.I, Feb. 2005.
6. S. Chiasson, P. van Oorschot, and R. Biddle. A usability study and critique of two password managers. In *USENIX Security*, 2006.
7. Clarke et al. The untrusted computer problem and camera-based authentication. In *Pervasive Computing*, volume 2414 of *LNCS*, 2002.
8. F-Secure. F-Secure virus descriptions: Cabir, June 2004.
9. F-Secure. F-Secure trojan information pages: Redbrowser.A, Mar. 2006.
10. E. W. Felten, D. Balfanz, D. Dean, and D. S. Wallach. Web spoofing: An Internet con game. In *National Information Systems Security Conference*, Oct. 1997.

11. A. Gostev and A. Shevchenko. Kaspersky security bulletin, January - June 2006: Malicious programs for mobile devices, Sept. 2006. `http://www.viruslist.com`.
12. N. Haller. The S/KEY one-time password system. RFC 1760, Feb. 1995.
13. G. Heiser. Secure embedded systems need microkernels. ;login:, Dec. 2005.
14. ICANN Security and Stability Advisory Committee. Domain name hijacking: Incidents, threats, risks, and remedial actions, July 2005. `http://www.icann.org`.
15. C. Jackson, D. Boneh, and J. Mitchell. Spyware resistant web authentication using virtual machines. Online manuscript. `http://crypto.stanford.edu/spyblock`.
16. King et al. SubVirt: Implementing malware with virtual machines. In *IEEE Symposium on Security and Privacy*, May 2006.
17. M. Mannan and P. C. van Oorschot. AVISPA test code for Mobile Password Authentication (MP-Auth). `http://www.scs.carleton.ca/~mmannan/mpauth`.
18. M. Mannan and P. C. van Oorschot. Using a personal device to strengthen password authentication from an untrusted computer (revised march 2007). Technical Report TR-07-11. `http://www.scs.carleton.ca/research/tech_reports/`.
19. N. B. Margolin, M. K. Wright, and B. N. Levine. Guardian: A framework for privacy control in untrusted environments, June 2004. Technical Report 04-37 (University of Massachusetts, Amherst).
20. J. M. McCune, A. Perrig, and M. K. Reiter. Bump in the Ether: A framework for securing sensitive user input. In *USENIX Annual Technical Conference*, 2006.
21. McCune et al. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *IEEE Symposium on Security and Privacy*, 2005.
22. J. Milletary. Technical trends in phishing attacks. US-CERT, Reading room article, `http://www.us-cert.gov/reading_room/phishing_trends0511.pdf`.
23. Mobile Antivirus Researchers Association. Analyzing the crossover virus: The first PC to Windows handheld cross-infector, 2006. `http://www.informit.com`.
24. Mobile Phone Work Group (MPWG). TCG mobile trusted module specification, Sept. 2006. Draft, version 0.9.
25. A. Moshchuk, T. Bragin, S. D. Gribble, and H. Levy. A crawler-based study of spyware in the web. In *Network and Distributed System Security (NDSS)*, 2006.
26. A. Oprea, D. Balfanz, G. Durfee, and D. Smetters. Securing a remote terminal application with a mobile trusted device. In *ACSAC*, 2004.
27. B. Parno, C. Kuo, and A. Perrig. Phoolproof phishing prevention. In *Financial Cryptography and Data Security (FC)*, volume 4107 of *LNCS*, 2006.
28. A. Perrig and D. Song. Hash visualization: A new technique to improve real-world security. In *Cryptographic Techniques and E-Commerce (CrypTEC)*, July 1999.
29. B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. Mitchell. Stronger password authentication using browser extensions. In *USENIX Security*, 2005.
30. J. Rutkowska. Introducing Blue Pill, 2006. Presented at SyScan, `http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.html`.
31. R. C. Stefan Berger, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn. vTPM: Virtualizing the trusted platform module. In *USENIX Security*, 2006.
32. Trend Micro. Mobile security. `http://www.trendmicro.com/en/products/mobile/tmms/evaluate/overview.htm`.
33. H. Tuch, G. Klein, and G. Heiser. OS verification — now! In *Hot Topics in Operating Systems*, June 2005.
34. P. C. van Oorschot. Message authentication by integrity with public corroboration. In *New Security Paradigms Workshop, (NSPW)*, Sept. 2005.
35. M. Wu, S. Garfinkel, and R. Miller. Secure web authentication with mobile phones. In *DIMACS Workshop on Usable Privacy and Security Systems*, July 2004.
36. Z. E. Ye, S. Smith, and D. Anthony. Trusted paths for browsers. *ACM Transactions on Information and System Security (TISSEC)*, 8(2), 2005.