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. Among the most significant current threats to online banking are keylogging and phishing. These attacks extract user identity and account information (e.g. userid, password) to be used later for unauthorized access to users' financial accounts. We propose a simple approach which cryptographically separates a user's long-term secret input from client (typically untrusted) 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, e.g., a password or PIN) 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 for online banking. We also provide a comprehensive survey of web authentication techniques – both proposed in the literature and/or developed in practice – that use an additional factor (e.g. a cellphone, PDA or hardware token) of authentication, and compare MP-Auth with these.

1 Introduction

Passwords enjoy ubiquitous use for online authentication. Although many more secure (typically also more complex and/or 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 one of the prime targets 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 (*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 [33] on a phishing website. In today's Internet environment, software keyloggers are typically installed on a user PC along with common malware and spyware [26]. 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 [2]. 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 *A*uthentication), is proposed primarily to protect a user's long-term password input through an untrusted (or rather, untrustworthy)

* Version: May 5, 2006. Contact author: `mmannan@scs.carleton.ca`.

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 (LCD) 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[1]), i.e., one free of malware. 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, i.e., tricking users to visit spoofed websites. Although phishing attacks have been known to the research community for at least 10 years (see the 1997 work of Felten et al. [14]), 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 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*) reported during a panel discussion at RSA 2006 [9] steal money from a user account in real-time after the user has logged in, instead of collecting usernames and passwords for later use. MP-Auth also protects against this, by providing transaction integrity through a transaction confirmation step.

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[2] when connected to the Internet. There are also now many anti-phishing proposals (e.g. [10], [33], [44]), and software "tools" designed to detect spoofed websites (e.g. eBay toolbar, SpoofGuard, Spoofstick, Netcraft toolbar). However, most of these solutions are susceptible to keylogging attacks.[3] On the other hand, several authentication schemes which use a trusted personal device, generally prevent keyloggers, but do not help against phishing attacks. Solutions that protect passwords from both keyloggers and phishing sites, and provide transaction security, are largely missing in today's Internet environment. MP-Auth is proposed as such a solution.

According to a January 2006 report [2], 92% of all phishing sites target online financial services, e.g., online banking and credit card transactions. MP-Auth primarily focuses on online banking. However, the protocol can be used for general web authentication systems as well as for ATMs.

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 in today's Internet environment.

1. Keylogging Protection. A client PC does not have access to long-term user secrets, which implies keyloggers 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, assuming reasonable encryption algorithms/parameters. Highly targeted phishing attacks (*spear phishing*) are also ineffective against MP-Auth.
3. Pharming Protection. In the unlikely event of domain name hijacking [20], MP-Auth does not reveal a user's long-term password to attackers. MP-Auth also protects passwords when the DNS cache of a client PC is poisoned.
4. Transaction Integrity. With the transaction confirmation step (see Section 2.2) 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.
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 ATM frauds (see Bond [5] for a list of ATM fraud cases).

We provide a comprehensive survey of related authentication schemes used in practice and/or proposed to date, and compare these to MP-Auth; this survey may be of independent interest. We analyze MP-Auth

[1] i.e., combining PDA and cellphone functionalities in the same hand-held device.
[2] The average time between attacks is 18 minutes as of May 3, 2006, reported in `http://isc.sans.org/`.
[3] PwdHash [33] can protect passwords from JavaScript keyloggers, but not software keyloggers on client PCs.

using the AVISPA (Automated Validation of Internet Security Protocols and Applications) [1] analysis tool; no attacks are found.

Differences with Existing Academic Work. Major differences of MP-Auth with other proposed mobile authentication techniques (e.g. [30], [8]) include the following (see detailed discussion in Section 5.2).

1. MP-Auth protects passwords from keylogging and phishing attacks, vs. just one or the other.
2. In contrast to authentication-only schemes, MP-Auth provides transaction integrity.
3. Several other solutions require a *trusted proxy*, introducing an extra deployment burden, and presenting an attractive target to determined attackers (providing access to many user accounts).
4. Fraudsters may increasingly target cellphones (i.e., the physical device) if long-term secrets are stored on them. MP-Auth avoids storing any secret on a cellphone.

Requirements and Drawbacks of MP-Auth. MP-Auth requires users to possess a malware-free[4] 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 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.[5] Users must compare easy-to-read words [18] or easily distinguishable images [32] generated from random binary strings. Usability of MP-Auth has not been tested; this is an architecture and state-of-the-art paper.

Organization. The sequel is organized as follows. The MP-Auth protocol, threat model and operational assumptions are discussed in Section 2. A brief analysis of MP-Auth messages, discussion on how MP-Auth prevents common attacks, 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. Related work, including commercial one-time password generators, and a number of web authentication techniques proposed in the literature, is discussed in Section 5. Section 6 concludes.

2 MP-Auth: A Protocol for Online Authentication

In this section, we describe the MP-Auth protocol, including threat model assumptions.

2.1 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 Section 2.2 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.

2.2 Protocol Steps in MP-Auth

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

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.

[4] There are known attacks against mobile devices [22], but the trustworthiness of such devices is currently more easily maintained than a PC, as they contain far less software; see Section 3.3.

[5] The first *crossover* virus has been reported [25] in February 2006.

U, M, B, S	User ID, a cellphone, a browser on the untrusted PC, and the server ID (e.g. URL), respectively.
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 long-term public key E_S.
X, Y	Concatenation of X and Y.
K_{BS}	Symmetric session (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 [18].

Table 1. Notation used in MP-Auth

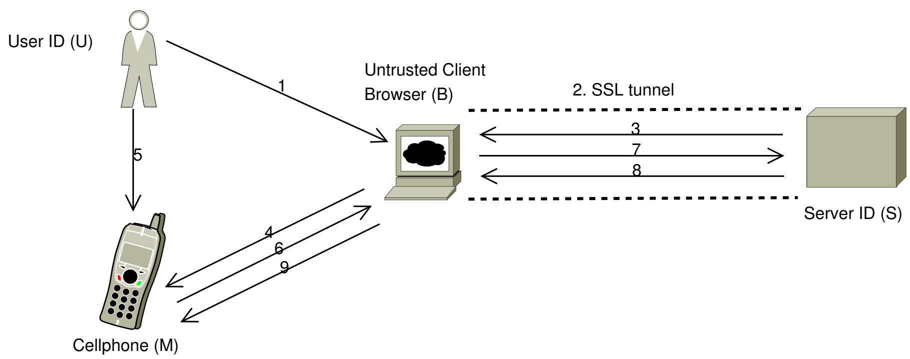

Fig. 1. MP-Auth protocol steps

3. S generates a random nonce R_S, and sends the following message to B.

$$B \leftarrow S: \; W, \text{ where } W = \{S, R_S\}_{K_{BS}} \tag{2.1}$$

4. B decrypts W and checks whether the received S is the same as the server identity with whom B established the SSL session earlier. On successful verification of S, B forwards W and S to M.

$$M \leftarrow B: \; W, S \tag{2.2}$$

   We describe an additional step called *session ID verification* (see below) in cases where the integrity of W must be protected.
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 U is, e.g., a bank card number).

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

7. B (through 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, U and W (and thereby 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 the verification result (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 a user's transaction or insert fraudulent transactions through the untrusted PC. To detect and prevent such transactions, MP-Auth requires transactions to be confirmed explicitly by U (through M). The following messages are exchanged for confirmation of a transaction with summary details T (R_{S1} is a server generated random nonce, used to prevent replay).

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

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

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 on U's behalf. 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 risk 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 W (when being forwarded to M), we add a session ID verification step after step 4. Both B and M compute a session ID $sid = v(W)$. B and M display sid to U. U proceeds only if both session IDs are the same. (For more on this, see *Parallel Session Attacks* in Section 3.2.) To minimize user errors, M shows a list of session IDs (one derived from W 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 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. postal mail), and through the password renewal web page, B and S establish an SSL session. The following message is forwarded from M to S (via B) during password renewal (let P_{old} and P_{new} are the old and new passwords respectively).

$$M \xrightarrow{X, \text{ where } X = \{R_M\}_{E_S}, \{U, P_{old}, P_{new}\}_{R_M}} B \xrightarrow{\{X\}_{K_{BS}}} S \tag{2.8}$$

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 (we caution that we have not tested the usability of any of these suggestions).

1. at a bank branch, preferably during an account setup (see Section 4 for usability issues).
2. through ATM interfaces, located at bank branches, hopefully no "fake" ATMs.
3. through a cellphone service (authenticated download) as data file transfer.

4. mailing CDs containing the public key (for web-only banks) to users; the public key is installed into a cellphone using a PC.

A challenge-response protocol or integrity cross-checks (using a different channel, e.g., see [39]) should be used to verify the public key installed on a cellphone, in addition to the above procedures (at least for the last three). 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* [32] or *public passwords* [17]).

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, and discuss several attacks that MP-Auth is resistant to. 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) [1] analysis tool, and found no attacks.[6] AVISPA test code for MP-Auth is provided in Appendix A along with discussion. We have not at this point carried out other formal analyses or security proofs for MP-Auth.

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 the encrypted value $W = \{S, R_S\}_{K_{BS}}$ to M. The server 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 at S. The use of W in message (2.4) chains together M, B and S; $f(W)$ is encrypted with R_M generated by M, R_S in W is generated by S, and K_{BS} is mutually established by B and S. Malware on the user PC may change W when B sends W to M. However, such a change will make the session IDs displayed on B and M different, and thus be detected, if session ID verification is implemented (assuming the PC display is trusted).

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; i.e., if a malicious S chooses R_S to be 0 (or other values), or W is replaced with a value of an attacker's choice, 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. If transaction integrity confirmation is used, 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 Unsuccessful Attacks Against MP-Auth

We list several potential attacks against MP-Auth, and discuss how MP-Auth prevents them. We also discuss some MP-Auth steps in greater detail, and further motivate various protocol components/steps.

a) Remote Desktop Attacks. A malicious browser B can collect message (2.4) and then deny access to U. B can use message (2.4) to login to S, and provide an attacker a remote desktop, such as a Virtual Network Computing (VNC) terminal, in real-time to the user PC. However, this attack will be detected if MP-Auth is implemented with the transaction integrity confirmation step.

[6] AVISPA is positioned as an industrial-strength technology for the analysis of large-scale Internet security-sensitive protocols and applications.

b) Session Hijacking Attacks. In a session hijacking attack, malware may take control of a user session after the user successfully establishes a session with the legitimate server; e.g., B may leak K_{BS} to malware. The malware may actively alter user transactions, or perform unauthorized transactions without immediately being noticed by the user. However, such attacks will be detected, if MP-Auth is implemented with the transaction integrity confirmation step.

c) Parallel Session Attacks. In a parallel session attack [7], messages from one protocol run are used to form another successful run by running two or more protocol instances simultaneously. Generally a parallel session attack may effectively be prevented through the proper chaining of protocol messages. However, in MP-Auth, there is no authentication between M and B, making such an attack possible even when protocol messages are linked correctly. An attack against MP-Auth, without session ID verification, is the following (see Fig. 2). When U launches B to visit S's site, malware from U's PC notifies a remote attacker A. A starts another session with S as U, and gets message (2.1) from S, which the attacker relays to U's PC. The malware on U's PC drops the message (2.1) intended for U when B attempts to send the message to M, and forwards the attacker's message to M instead. The malware then relays back U's response (i.e., from M) to A. Now A can login as U for the current session, although A is unable to learn P. However, this attack fails against MP-Auth if session ID verification is used, because the session IDs displayed on B and M will be different, and assumed to be detected by the user. If transaction integrity confirmation is implemented, such parallel session attacks become meaningless (view-only) even without session ID verification.

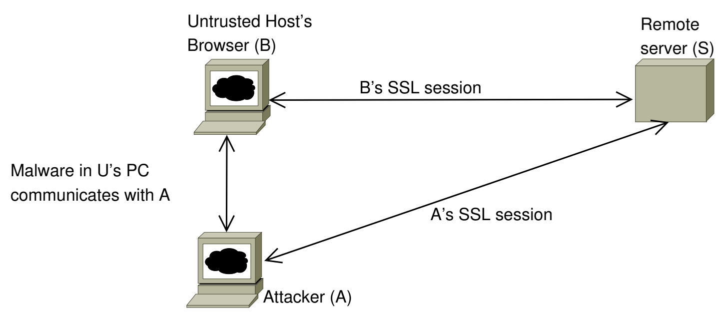

Fig. 2. Setup for a parallel session attack

d) Man in the Middle Attacks. We consider here a man-in-the-middle (MITM) attack where the attacker sits between a user's browser and a bank website, in an attempt to establish a session with the website as the user (see Fig. 3). Here we consider only a typical MITM attack, where the attacker lures a user to a phishing site (e.g. via domain name hijacking [20]).

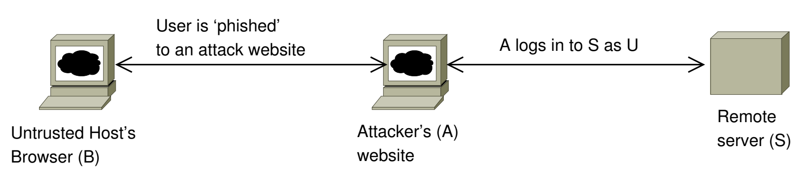

Fig. 3. Setup for a man-in-the-middle attack

In this attack, the user visits a phishing website assuming that to be the legitimate bank website. Attacker A establishes an SSL session with U posing as S, and A opens up another SSL session with S posing as U; assume the established SSL session keys are K_{BA} and K_{AS} respectively. A receives $\{S, R_S\}_{K_{AS}}$ from S in message (2.1) which A forwards to B to get the response message (2.4). A ideally would want to replace $\{A, R_S\}_{K_{BA}}$ with $\{S, R_S\}_{K_{AS}}$, otherwise the response message (2.4) from M cannot be used to login as U; recall that, M puts hash of the content of message (2.1) encrypted inside message (2.4). However, such a replacement will result in a failure on verifying server ID S when message (2.1) is decrypted by B, and B will terminate the session. Hence, a typical MITM attack will apparently fail against MP-Auth. However, if B is compromised, a 'successful' MITM attack becomes meaningless (view-only) when MP-Auth is implemented with transaction integrity confirmation.

3.3 Remaining Attacks Against MP-Auth

Although MP-Auth apparently protects user passwords from malware installed on a PC or any phishing website, here we discuss some other possible attacks against MP-Auth. These attacks, if successful, may reveal a user's plaintext password to attackers.

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 (e.g. [22]) as high-end cellphones (smart phones) contain millions of lines of code. Worms (e.g. Cabir [12]) are designed to spread in smart phones by exploiting vulnerabilities in embedded operating systems. However, currently cellphones remain far more trustworthy than PCs, thus motivating our proposal. In the future, as mobile devices increasingly contain much more software, this requirement of trustworthy cellphones becomes problematic, and their use for sensitive purposes such as online banking makes them a more attractive target. Limited functionality devices (with less software, implying more trustworthy) may then provide an option for use with MP-Auth. Note that, 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 other two-factor authentication proposals. Another possibility of restricting mobile malware may be the use of micro-kernels [19], formally verifiable OS kernels [38], or trusted virtual machine monitors (TVMM) [15] on cellphones to restore a trustworthy application environment.

b) Private Key Disclosure. It would be disastrous if the private key of a bank is compromised. This would require, e.g., that the bank generate and distribute a new public key. However, this threat also exists for currently deployed SSL (server site) certificates, and root keys present in current browsers.

c) Shoulder Surfing Attacks. A nearby attacker may observe (*shoulder surf*) while a user enters a password to a mobile device. Video recorders or cellphones with a video recording feature can also easily record user passwords/PINs in a public location, e.g., in an ATM booth. MP-Auth does not stop such attackers. Methods resilient against shoulder surfing have been proposed (e.g. [40], [34]), and may be integrated with MP-Auth albeit their practical viability remains an open question.

d) 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 retrive 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 [33]).

e) 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 recent (April 25, 2006) "mixed" phishing attack,[7] 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 banking website, and enters into the cellphone the userid and password for the site (step 5, see Section 2.2), if MP-Auth session ID verification is successful (assuming session ID verification is implemented). We also assume that typing a userid and password on a cellphone keypad is acceptable in terms of usability, as many users are 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.

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

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 we assume, based on results from previous implementations (e.g. [30]), poses no practical barrier. (Indeed in our application, the operation is a relatively cheap RSA encryption with short public exponent, rather than a more expensive RSA decryption with private key.) We now discuss other usability and deployment aspects which may favor MP-Auth (see also Section 5.1 (e)).

1. As it appears from the current trend in online banking (see Section 5.1), users are increasingly required to use two-factor authentication (e.g. with a separate device such as a passcode generator) for login. Hence using (an existing) cellphone or related device instead of a specialized device for online banking relieves users from carrying an extra device. Also, a user might otherwise require multiple hardware tokens (e.g. SecurID, Chip and PIN card) for accessing different online accounts (from different banks).
2. The usability of four login techniques has been studied by Wu et al. [42] – 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 [42].
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.

Although we have not tested MP-Auth for usability, the above discussion suggests that compared to available two-factor authentication methods (see Section 5.1), MP-Auth maybe as usable or better. However, we hesitate to make strong statements without usability tests (e.g. see [6]).

5 Survey of Related Work

In this section, we summarize and provide extended discussion of related online authentication methods used in practice or proposed in the literature, and compare MP-Auth with these techniques.

5.1 Online Authentication Methods

We first discuss several online authentication methods commonly used (or proposed for use) by banks, and briefly discuss their security.

a) Password-only Authentication. Most bank websites authenticate customers using only a password over an SSL connection. This is susceptible to keyloggers and phishing. Banks' reliance on SSL certificates does not stop attackers. Attackers have used certificates – both self-signed, and real third-party signed certificates for *sound-alike* domains, e.g., `visa-secure.com` – to cause the SSL lock on phishing websites to be displayed. In 2005, over 450 phishing websites were reported to deploy SSL [28].

In a cross-site/cross-frame scripting attack, vulnerable website software is exploited to display malicious (phishing) contents within the website, making such attacks almost transparent to users. Past vulnerable websites include Charter One Bank, MasterCard, Barclays and Natwest [45]. In a March 2006 phishing attack, attackers broke into web servers of three Florida-based banks, and redirected the banks' customers to phishing websites.[8] In another high-profile phishing attack, attackers manipulated a U.S. government website to forward users to phishing websites.[9]

Reliance on SSL itself also leads to problems. For example, only one in 300 customers of a New Zealand bank [28] chose to abandon the SSL session upon a browser warning indicating an expired SSL site certificate; the bank accidentally allowed a certificate to expire for a period of 12 hours. A user study by Dhamija et al. [11] also notes that standard (visual) security indicators on websites are ineffective for a significant portion of users; over 90% participants were fooled by phishing websites in the study.

The above suggests password-only web authentication over SSL is inadequate in today's Internet environment. This is motivating financial organizations towards two-factor authentication methods.

b) Two-factor Authentication. Traditionally, authentication schemes have relied on one or more of three factors: something a user *knows* (e.g. a password), something a user *has* (e.g. a bank card), and something a user *is* (e.g. biometric characteristics). Properly designed authentication schemes that depend on more than one factor are more reliable than single-factor schemes. Note that the authentication scheme used in ATMs through a bank card and PIN is two-factor; but, an online banking authentication scheme that requires a user's bank card number (not necessarily the card itself) and a password is single-factor, i.e., both are something *known*. As a step toward multi-factor authentication, banks are providing users with devices like one-time password generators, to use along with passwords for online banking, thus making the authentication scheme rely on two independent factors. Examples of two-factor authentication in practice are given below.

1. Several European banks attempt to secure online banking through e.g., passcode generators.[10]
2. U.S. federal regulators have provided guidelines for banks to implement two-factor authentication by the end of 2006 for online banking [13].
3. The Association of Payment and Clearing Systems (APACS) in the U.K. is developing a standard[11] for online and telephone banking authentication. Most major U.K. banks and credit-card companies are members of APACS. The standard provides users a device to generate one-time passwords using a chip card and PIN. The one-time password is used along with a user's regular password.

Two-factor web authentication methods may make the collection of passwords less useful to attackers and thus help restrict phishing attacks. However, these methods raise deployment and usability issues, e.g., cost of the token, requirement to carry the token. Also malware on a client PC can record the device-generated secret (which a user inputs directly to a browser), and log on to the bank website before the actual user. This is recognized as a classic man-in-the-middle (MITM) attack [35].

In an interesting real attack [36] against a one-time password scheme implemented by a Finnish bank, the bank provided users a scratch sheet containing a certain number of one-time passwords. By setting up several phishing sites, attackers persuaded users to give out a sequence of one-time passwords in addition to their regular passwords. This attack is made more difficult if one-time passwords expire after a short while (e.g. 30 to 60 seconds in SecurID); then the collected one-time passwords must be used within a brief period of time from a user's login attempt. Such time-based passcode generators, e.g., SecurID, typically have time synchronization problems between a client device and the server [43]. Other security issues of such devices (e.g. [41], [27], [4], [31]) are not directly relevant to our discussion; we assume that any weaknesses could be repaired by superior algorithms or implementations overtime, albeit with the usual practical challenges, e.g., backwards compatibility.

Note that, even when a one-time password is used along with a user's (long-term) regular password, gathering long-term passwords may be still be of offline use to an attacker. For example, if flaws are found in a one-time key generator algorithm (e.g., *differential adaptive chosen plaintext* attack [4]) by which attackers can generate one-time keys without getting hold of the hardware token, keylogging attacks to collect user passwords appear very useful.

[8] `http://news.netcraft.com/archives/2006/03/27/phishers_hack_bank_sites_redirect_customers.html`
[9] `http://www.eweek.com/article2/0,1895,1894746,00.asp`
[10] A list of bank websites that use SSL login and/or two-factor authentication is available at `https://www.securewebbank.com/loginssluse.html`.
[11] `http://www.chipandpin.co.uk/`

Instead of gathering passwords, attackers can simply steal money from user accounts in real-time, immediately after a user completes authentication [9]. Therefore, transaction security, as possible in MP-Auth, becomes critical to restrict such attacks.

c) Transaction Security and Complimentary Mechanisms. To protect important transactions, and make users better able to detect break-ins to their accounts, some banks have deployed security techniques which are generally complementary to authentication schemes. Examples include:

1. Two New Zealand banks require online users to enter a secret from a cellphone (sent as an SMS message to the phone) for transfers over $2500 from one account to another [37].
2. Customers of the Commonwealth Bank of Australia[12] must answer (pre-established) identification questions when performing sensitive transactions. Email alerts are sent to users to confirm when users' personal details have been changed, or modifications to user accounts are made.
3. Bank of America uses SiteKey[13] to strengthen online authentication. If a user PC is recognized by the bank, a secret pre-shared SiteKey picture is displayed; upon successful verification of the SiteKey picture, the user enters her password. A confirmation question is asked if the user PC is not recognized, and the SiteKey picture is displayed when the user answers the question correctly. The SiteKey picture provides evidence that the user is entering her password to the correct website.
4. Users can view a real-time transaction summary of the current session on the CIBC (Canada) online banking website. Also, the date and time of a user's last login are displayed on the website.

In principle, the above mechanisms (as well as MP-Auth's transaction integrity confirmation) are similar to integrity cross-checks by a second channel [39]. These appear to be effective only against high-impact online frauds. Attackers may be able to defeat some of these techniques; e.g., if a bank requires SMS verification on large transactions, attackers can commit several relatively small transactions (e.g. $10 instead of $1000) to avoid the verification step.

d) Using a Cellphone Alone for Important Internet Services. Some [16] believe cellphones have the potential to replace commodity PCs entirely. One proposed solution to keyloggers is to perform all critical work through a cellphone browser, or through a PDA. However, a combination of the following usability and security issues may restrict such proposals being widely deployed.

1. The display area of a cellphone/PDA is much smaller than a PC, limiting usability for web browsing.
2. Users may still reveal passwords to phishing sites controlled by malicious parties (through e.g., domain name hijacking [20]). Thus even a trusted browser in a trusted device may not stop phishing attacks; i.e., such a setup may allow a 'secure' pipe directly to phishing sites.
3. In many parts of the world, airtime costs money. So Internet browsing through a mobile network remains, at least presently, far more expensive than wire-line Internet connections.

e) Comparing MP-Auth with Existing Online Authentication Methods. 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 2.2) 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.)

5.2 Academic Work

Here we summarize selected academic work on authentication from an untrusted PC using a mobile device. We also compare MP-Auth to these proposals in terms of technical merits and usability.

a) Splitting Trust Paradigm. In 1999, Balfanz and Felten [3] proposed a scheme to deliver smart card functionality through a PalmPilot. They introduced the *splitting trust* paradigm to split an application

[12] `http://www.commbank.com.au/Netbank/faq/security.asp`

[13] `http://www.bankofamerica.com/privacy/passmark/`

between a small (in size and processing power) 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.

b) Phoolproof Phishing Prevention. Parno, Kuo and Perrig [30] proposed 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 (see Fig. 4), a user provides userid and password to a website (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).

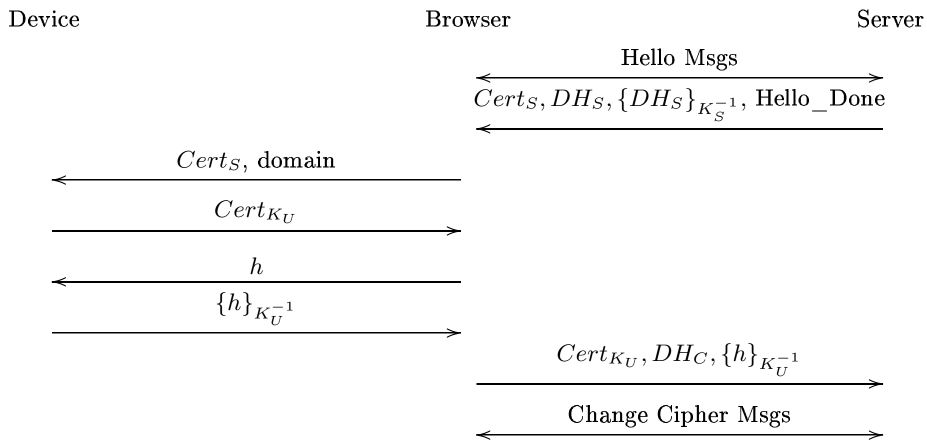

Fig. 4. Phoolproof login process.

In Figure 4, DH_S, DH_C represent the Diffie-Hellman public key parameters for the server and client browser respectively, and h is a secure hash of all previous SSL handshake messages of the current session. As noted [30], attackers may hijack account setup or (user) public key re-establishment. 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) [24] techniques are proposed to secure local Bluetooth channels, requiring users to take snapshots using a camera-phone, apparently increasing complexity to users. If malware on a PC can replace h (when the browser attempts to send h to the cellphone) with an h value from an attacker, the attacker can login as the user (recall *Parallel Session Attacks* in Section 3.2). 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.

c) SpyBlock. Jackson, Boneh and Mitchell propose SpyBlock [21] to provide *spyware-resistant* web authentication using a virtual machine monitor (VMM). The SpyBlock authentication agent runs on a host OS (assumed to be trusted), and user applications including a web browser with a SpyBlock browser helper run inside a guest VMM (assumed to be untrusted) on the trusted host OS. See Figure 5.

A user authenticates to a website with the help of the SpyBlock authentication agent. The site password is given only to the authentication agent which supports several authentication techniques, e.g., password hashing, strong password authentication, transaction integrity confirmation. The authentication agent provides a *trusted* path to the user through a pre-shared secret picture.

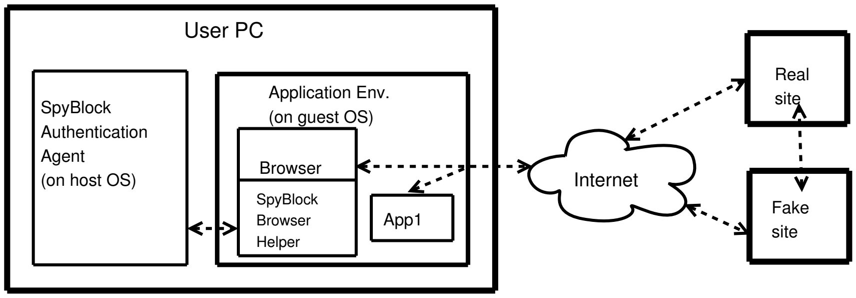

Fig. 5. SpyBlock setup

Although SpyBlock does not require an additional hardware device (e.g. a cellphone), a VMM must be installed on top of a host OS; the current reality is that most users do not use any VMM. Also, users must know when they are communicating with the authentication agent; user interface design in such a setting appears quite challenging. Another assumption in SpyBlock is that the host OS is *trusted*. In reality, maintaining trustworthiness of any current consumer OS is very difficult (which is in part why secure web authentication is so complex).

d) Three-party Secure Remote Terminal Protocol. Oprea et al. [29] proposed a three-party protocol (see Fig. 6) to provide secure access to a home computer from an untrusted public terminal. A trusted device (PDA) is used to delegate temporary credentials of a user to an untrusted public computer, without revealing any long-term secret to the untrusted terminal. Two SSL connections are established in the protocol: one from the trusted PDA and another from the untrusted terminal to the home PC using a modified Virtual Network Computing (VNC) system. The PDA authenticates normally (using a password) to the home PC, and forwards temporary secret keys to the untrusted terminal. A user can control how much information from the home PC is displayed to the untrusted PC. Control messages to the home VNC, e.g., mouse and keyboard events are only sent from the PDA.

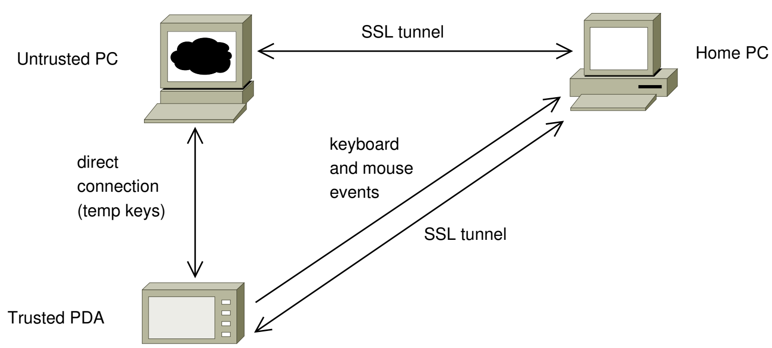

Fig. 6. Three-party VNC protocol

Although this protocol safeguards user passwords, it does so when users access a PC (or application) that they control, e.g., a home PC. MP-Auth is aimed to protect user passwords in a more general Internet setting, i.e., when users access a hosted service (e.g. an online banking website). In the VNC protocol, the trusted device must have SSL capabilities, and is required to maintain a separate SSL channel from the PDA to the home PC; MP-Auth requires neither of these.

e) Camera-based Authentication. Clarke et al. [8] proposed a technique using camera-phones for authenticating visual information (forwarded by a trusted service) in an untrusted PC. This method verifies message authenticity and integrity for an entire user session; i.e., it authenticates contents displayed on a PC screen for every web page or only critical pages in a user session. A small area on the bottom of a PC screen is used to transmit security parameters (e.g. a nonce, a one-time password, a MAC) as an image, with a strip of random-looking data. Figure 7 outlines the proposed protocol.

To access a service from the Internet through an untrusted PC, this scheme requires a *trusted proxy*. A user's long-term keys are stored on the camera-phone, protected by a PIN or biometric measurement.

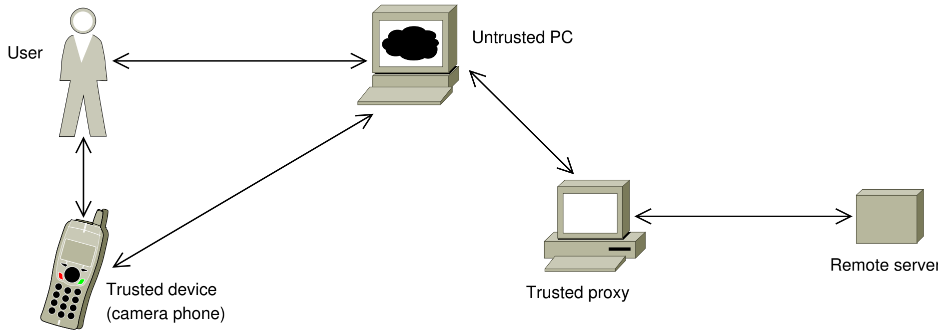

Fig. 7. Camera-based authentication

With a stolen phone, an attacker may successfully impersonate the user or retrieve the stored long-term keys from the phone. Camera-based authentication also creates a much different user experience: users are expected to take snapshots and visually verify (cross-check) images in terms of colors and shades. A calibration phase may also be required to construct a mapping between PC screen pixels and camera pixels (in one implementation, reported to take about 10 seconds). It attempts to authenticate contents of a visual display, which is apparently useful in a sense that we can verify what is displayed on the screen. The transaction confirmation step in MP-Auth provides effectively similar protection for online banking, without issues such as screen snapshot capture or pixel calibration.

f) Secure Web Authentication with Cellphones. Figure 8 shows the secure web authentication proposed by Wu et al. [42]. User credentials (userid, password, mobile number etc.) are stored on a trusted proxy server. The protocol involves the following steps (see Fig. 8 for symbol definitions).

1. U launches a web browser at K, and goes to T's site.
2. U types her userid and K sends it to T.
3. T chooses a random session name, and sends it to K.
4. T sends the same session name to M as an SMS message.
5. U checks the displayed session name at K.
6. U verifies the session name at M.
7. If session names match, user accepts the session.
8. If U accepts the session, then T uses U's stored credentials to login to R, and works as a web proxy.

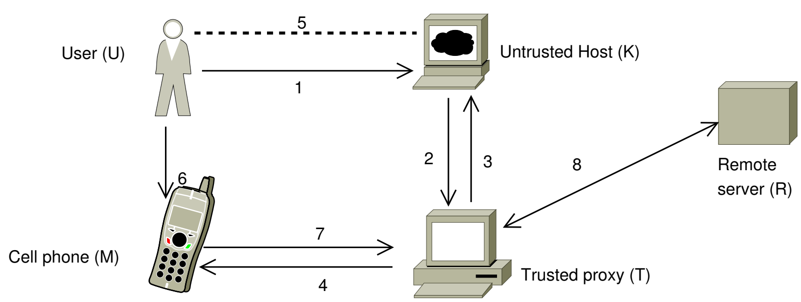

Fig. 8. Web authentication with a cellphone

This protocol pursues similar goals as ours, but requires a *trusted proxy*, which if compromised, may readily expose user credentials to attackers. A well-behaved proxy may also be tricked to access a service on behalf of a user. Hence the proxy may become a prime target of attacks. Also, losing the cellphone may be disastrous, as anyone can access the trusted proxy using the phone, at least temporarily.

g) Guardian: A Framework for Privacy Control. The Guardian [23] framework has been designed with an elaborate threat model in mind. Its focus is to protect privacy of a mobile user,[14] including securing long-term user passwords and protecting sensitive information, e.g., personal data from being recorded (to prevent identity profiling). Guardian works as a personal firewall but placed on a trusted PDA. In effect, the PDA acts as a *portable privacy proxy*. See Fig. 9.

[14] A user who uses several different public terminals to access critical online services, e.g., banking.

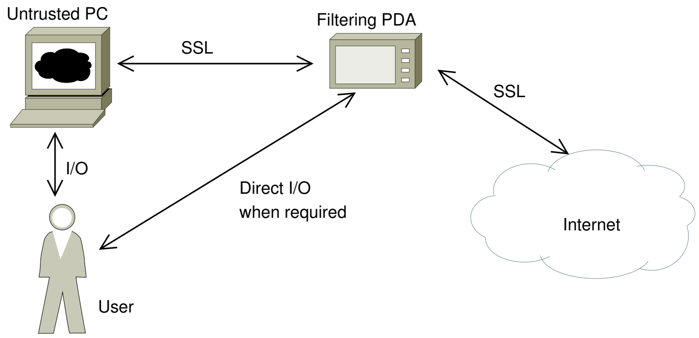

Fig. 9. Guardian setup

Guardian keeps passwords and other privacy sensitive information out of the reach of keyloggers and other malware installed on an untrusted PC. However, phishing attacks still may succeed. Guardian attempts to manage a large set of sensitive user details, e.g., PKI certificates, SSL connections, and cookies as well as real-time content filtering. Thus its implementation appears to be complex, and requires intelligent processing from the PDA. Although MP-Auth does not protect user privacy, it provides protection against both keyloggers and phishing (for online banking), and apparently much simpler than Guardian.

6 Concluding Remarks

We have proposed MP-Auth, a protocol for web authentication which is resilient to keyloggers (and other malware including rootkits) and phishing websites, and provides transaction integrity. 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 Internet financial transactions (e.g. online banking, credit card transactions). 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 (i.e. userid and password). MP-Auth is designed to prevent such attacks. 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 3.3, may well remain viable.

Users often input reusable critical identity information to a PC other than userid/password, e.g., passport number (Amtrak website), social security number, driver's licence number, birth date, 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 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 tested for usability. As discussed in Section 4, computational costs pose no barrier. We encourage the security community to pursue alternate proposals for password-based online authentication which simultaneously address phishing, keylogging and rootkits.

Acknowledgements

We thank Paul H. Drielsma for help with AVISPA [1]. The first author is supported in part by the Public Safety and Emergency Preparedness Canada (PSEPC) Program. 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. A. 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`.
2. Anti-Phishing Working Group. Phishing Activity Trends Report, January, 2006. `http://www.antiphishing.org/reports/apwg_report_jan_2006.pdf`.
3. D. Balfanz and E. Felten. Hand-held computers can be better smart cards. In *USENIX Security*, 1999.
4. A. Biryukov, J. Lano, and B. Preneel. Cryptanalysis of the alleged SecurID hash function. In *Selected Areas in Cryptography (SAC)*, volume 3006 of *LNCS*, 2003.
5. M. Bond. Phantom withdrawals: On-line resources for victims of ATM fraud. `http://www.phantomwithdrawals.com/phantoms.html`.
6. S. Chiasson, P. van Oorschot, and R. Biddle. A usability study and critique of two password managers. In *USENIX Security*, 2006.
7. J. Clark and J. Jacob. Attacking authentication protocols. *High Integrity Systems*, 1(5), 1996.
8. D. E. Clarke, B. Gassend, T. Kotwal, M. Burnside, M. van Dijk, S. Devadas, and R. L. Rivest. The untrusted computer problem and camera-based authentication. In *Pervasive Computing*, volume 2414 of *LNCS*, 2002.
9. CNET News Staff. New trojans plunder bank accounts, Feb. 2006. `http://www.news.com/`.
10. R. Dhamija and J. D. Tygar. The battle against phishing: Dynamic Security Skins. In *Symposium on Usable privacy and security (SOUPS)*, 2005.
11. R. Dhamija, J. D. Tygar, and M. Hearst. Why phishing works. In *Conference on Human Factors in Computing Systems (CHI)*, Apr. 2006.
12. F-Secure. F-Secure virus descriptions: Cabir, June 2004. `http://www.f-secure.com/v-descs/cabir.shtml`.
13. Federal Financial Institutions Examination Council (FFIEC). FFIEC guidance: Authentication in an Internet banking environment, Oct. 2005. `http://www.fdic.gov/news/news/financial/2005/fil10305.html`.
14. 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.
15. T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A virtual machine-based platform for trusted computing. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2003.
16. P. Greenspun. Mobile phone as home computer, Sept. 2005. `http://philip.greenspun.com/business/mobile-phone-as-home-computer`.
17. S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. *ACM Transactions on Information and Systems Security*, 2(3), 1999.
18. N. Haller. The S/KEY one-time password system. RFC 1760 (Informational), Feb. 1995.
19. G. Heiser. Secure embedded systems need microkernels. ;login: The USENIX Magazine, Dec. 2005.
20. ICANN Security and Stability Advisory Committee. Domain name hijacking: Incidents, threats, risks, and remedial actions, July 2005. `http://www.icann.org/announcements/hijacking-report-12jul05.pdf`.
21. C. Jackson, D. Boneh, and J. Mitchell. Spyware resistant web authentication using virtual machines. Online manuscript. `http://crypto.stanford.edu/spyblock/spyblock.pdf`.
22. Kaspersky Lab. Trojan targets mobiles phones running Java applications, Feb. 2006. `http://www.kaspersky.com/news?id=180984542`.
23. 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. `http://prisms.cs.umass.edu/brian/pubs/margolin.wright.guardian.pdf`.
24. J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *IEEE Symposium on Security and Privacy*, May 2005.
25. Mobile Antivirus Researchers Association. Analyzing the crossover virus: The first PC to Windows handheld cross-infector, Mar. 2006. News article. `http://www.informit.com/articles/article.asp?p=458169&rl=1`.
26. 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)*, Feb. 2006.
27. Mudge and Kingpin. Initial cryptanalysis of the RSA SecurID algorithm, 2001. White paper. `http://www.linuxsecurity.com/resource_files/cryptography/initial_securid_analysis.pdf`.
28. Netcraft. More than 450 phishing attacks used SSL in 2005, Dec. 2005. `http://news.netcraft.com`.
29. A. Oprea, D. Balfanz, G. Durfee, and D. Smetters. Securing a remote terminal application with a mobile trusted device. In *Annual Computer Security Applications Conference (ACSAC)*, 2004.
30. B. Parno, C. Kuo, and A. Perrig. Phoolproof phishing prevention. In *Financial Cryptography (FC)*, LNCS (to appear), 2006.
31. PeiterZ@silence.secnet.com. Weaknesses in SecurID. White paper. `http://www.tux.org/pub/security/secnet/papers/secureid.pdf`.

32. A. Perrig and D. Song. Hash visualization: A new technique to improve real-world security. In *Workshop on Cryptographic Techniques and E-Commerce (CrypTEC)*, July 1999.
33. B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. Mitchell. Stronger password authentication using browser extensions. In *USENIX Security*, 2005.
34. V. Roth, K. Richter, and R. Freidinger. A PIN-entry method resilient against shoulder surfing. In *ACM Computer and communications Security (CCS)*, 2004.
35. B. Schneier. Two-factor authentication: Too little, too late. *Comm. of the ACM*, 48(4):136, Apr. 2005.
36. The Register Staff. Phishing attack targets one-time passwords, Oct. 2005. `http://www.theregister.co.uk/2005/10/12/outlaw_phishing/`.
37. The Sydney Morning Herald Staff. NZ bank adds security online, Nov. 2004. `http://www.smh.com.au/`.
38. H. Tuch, G. Klein, and G. Heiser. OS verification — now! In *Hot Topics in Operating Systems*, June 2005.
39. P. C. van Oorschot. Message authentication by integrity with public corroboration. In *New Security Paradigms Workshop, (NSPW)*, Sept. 2005.
40. D. Weinshall. Cognitive authentication schemes safe against spyware (short paper). In *IEEE Symposium on Security and Privacy*, May 2006.
41. I. Wiener. Sample SecurID token emulator with token secret import, 2000. BugTraq post. `http://archives.neohapsis.com/archives/bugtraq/2000-12/0428.html`.
42. M. Wu, S. Garfinkel, and R. Miller. Secure web authentication with mobile phones. In *DIMACS Workshop on Usable Privacy and Security Systems*, July 2004. `http://sow.csail.mit.edu/2003/proceedings/Wu.pdf`. See also the abstract at `http://dimacs.rutgers.edu/Workshops/Tools/abstract-wu-garfinkel-miller.pdf`.
43. G. G. Xie, C. E. Irvine, and T. E. Levin. Quantifying effect of network latency and clock drift on time-driven key sequencing. In *International Conference on Distributed Computing Systems (ICDCSW)*, 2002.
44. E. Ye and S. Smith. Trusted paths for browsers: An open-source solution to web spoofing, Feb. 2002. Technical Report TR2002-418, Dartmouth College. `http://www.cs.dartmouth.edu/~pkilab/papers/tr418.pdf`.
45. ZapTheDingbat. Has MasterCard gone on a phishing trip, leaving the back door wide open?, June 2004. `http://www.zapthedingbat.com/security/scriptinjection/`.

A AVISPA Test Code

Protocol: MP-Auth

As noted in Section 3, we include here results on our AVISPA [1] analysis of an idealized version (see below) of the MP-Auth protocol from Section 2.2.

Protocol Purpose

Authentication and key exchange between a mobile device M and a remote server S. More specifically, goals are (see Section 2.2, Table 1 for notation):

- M and S achieve mutual authentication (using P and E_S)
- M and S establish a secret (symmetric) session key for later use in encryption

How We Tested Using AVISPA

We used AVISPA Web interface available at `http://www.avispa-project.org/web-interface/`. We copied the HLPSL code (below) to the Web interface, and ran the relevant tests. Applicable tests to MP-Auth are: On the Fly Model Checker (OFMC), Constraint Logic-based Attack Searcher (CL-AtSe), and SAT-based Model Checker (SATMC). The Tree Automata based on Automatic Approximations for Analysis of Security Protocols (TA4SP) results are omitted from the AVISPA output below as the TA4SP back-end was not supported for our setup.

Idealization of MP-Auth

In MP-Auth, the browser B acts like a relaying party between M and S during the authentication and key exchange phase. Therefore B was removed from our idealized HLPSL model (and thus also, the SSL encryption between B and S). Also, the human user U was merged with M, as U only provides the password P to M. Hence the idealized MP-Auth is a two-party protocol, which is much simpler to analyze for AVISPA back-end protocol analyzers. As we have omitted party B, session ID verification is not required. The transaction integrity confirmation messages use K_{MS} established in the authentication phase. The confirmation messages have not been included in our model; we assume the secrecy of K_{MS} implicitly protects those messages. The idealized version of MP-Auth is given below.

```
M <- S: Rs
M -> S: {Rm}_Es, {f(Rs), M, P}_Kms, where Kms = f(Rs, Rm)
M <- S: {f(Rm)}_Kms
```

Results of the AVISPA Tests

No attacks have been reported by AVISPA on the idealized protocol. Results from the AVISPA back-end protocol analyzers are given below.

OFMC.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/avispa/web-interface-computation/./tempdir/workfileP2NEkh.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 2.58s
  visitedNodes: 798 nodes
  depth: 10 plies
```

CL-AtSe.

```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  /home/avispa/web-interface-computation/./tempdir/workfileP2NEkh.if
GOAL
  As Specified
BACKEND
  CL-AtSe
STATISTICS
  Analysed   : 5548 states
  Reachable  : 3529 states
  Translation: 0.01 seconds
  Computation: 0.14 seconds
```

SATMC.

```
SUMMARY
  SAFE
DETAILS
  STRONGLY_TYPED_MODEL
```

```
  BOUNDED_NUMBER_OF_SESSIONS
  BOUNDED_SEARCH_DEPTH
  BOUNDED_MESSAGE_DEPTH
PROTOCOL
  workfileP2NEkh.if
GOAL
  %% see the HLPSL specification..
BACKEND
  SATMC
COMMENTS
STATISTICS
  attackFound              false     boolean
  upperBoundReached        true      boolean
  graphLeveledOff          4         steps
  satSolver                zchaff    solver
  maxStepsNumber           11        steps
  stepsNumber              5         steps
  atomsNumber              1196      atoms
  clausesNumber            5705      clauses
  encodingTime             1.12      seconds
  solvingTime              0.1       seconds
  if2sateCompilationTime   0.21      seconds
ATTACK TRACE
  %% no attacks have been found..
```

HLPSL Specification

```
role mobile (M, S: agent,
      Es: public_key,
      F, KeyGen: hash_func,
      P: text,
      SND, RCV: channel (dy)) played_by M def=

  local State : nat,
  Rm, Rs: text,
  Kms: message

  init State := 1

  transition
    2.  State  = 1 /\ RCV(Rs') =|>
        State':= 3 /\ Rm' := new()
                   /\ Kms':= KeyGen(Rs'.Rm')
                   /\ SND({Rm'}_Es.{F(Rs').M.P}_Kms')
                   /\ witness(M,S,rm,Rm')
                   /\ secret(Kms', sec_kms1, {M,S})

    3.  State  = 3 /\ RCV({F(Rm)}_Kms) =|>
        State':= 5 /\ request(M,S,rs,Rs)
end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role server(S: agent,
      Es: public_key,
      F, KeyGen: hash_func,
      Agents: (agent.text) set,
      SND, RCV: channel (dy)) played_by S def=
```

```
  local State : nat,
  Rm, Rs, P: text,
  Kms: message,
  M: agent

  init State := 0

  transition
    1.  State  = 0 /\ RCV(start) =|>
        State':= 2 /\ Rs' := new()
                   /\ SND(Rs')

    2. State  = 2  /\ RCV({Rm'}_Es.{F(Rs).M'.P'}_KeyGen(Rs.Rm'))
                   /\ in(M'.P', Agents) =|>
       State':= 4  /\ Kms' := KeyGen(Rs.Rm')
                   /\ SND({F(Rm')}_Kms')
                   /\ secret(Kms', sec_kms2, {M',S})
                   /\ request(S,M',rm,Rm')
                   /\ witness(S,M',rs,Rs)
end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role session(M, S: agent,
       Es: public_key,
       F, KeyGen: hash_func,
       P: text,
       Agents: (agent.text) set) def=

  local SS, RS, SM, RM: channel (dy)

  composition
     mobile (M,S,Es,F,KeyGen,P,SM,RM)
  /\ server (S,Es,F,KeyGen,Agents,SS,RS)
end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role environment() def=

    local Agents: (agent.text) set
    const m, s: agent,
    es: public_key,
    f, keygen: hash_func,
    rm, rs, sec_kms1, sec_kms2 : protocol_id,
    pm, pi: text

    init Agents := {m.pm, i.pi}
    intruder_knowledge = {m,s,f,keygen,pi,es,rs}

    composition
        session(m,s,es,f,keygen,pm,Agents)
     /\ session(m,s,es,f,keygen,pm,Agents)
     /\ session(i,s,es,f,keygen,pi,Agents)
end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

goal
```

```
  secrecy_of sec_kms1, sec_kms2
  authentication_on rm
  authentication_on rs

end goal

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

environment()
```