Phoolproof Phishing Prevention

Bryan Parno Cynthia Kuo Adrian Perrig

Carnegie Mellon University

Abstract. Phishing, or web spoofing, is a growing problem: the Anti-Phishing Working Group (APWG) received almost 14,000 unique phishing reports in August 2005, a 56% jump over the number of reports in December 2004 [3]. For financial institutions, phishing is a particularly insidious problem, since trust forms the foundation for customer relationships, and phishing attacks undermine confidence in an institution.

Phishing attacks succeed by exploiting a user's inability to distinguish legitimate sites from spoofed sites. Most prior research focuses on assisting the user in making this distinction; however, users must make the right security decision every time. Unfortunately, humans are ill-suited for performing the security checks necessary for secure site identification, and a single mistake may result in a total compromise of the user's online account. Fundamentally, users should be authenticated using information that they cannot readily reveal to malicious parties. Placing less reliance on the user during the authentication process will enhance security and eliminate many forms of fraud.

We propose using a trusted device to perform mutual authentication that eliminates reliance on perfect user behavior, thwarts Man-in-the-Middle attacks after setup, and protects a user's account even in the presence of keyloggers and most forms of spyware. We demonstrate the practicality of our system with a prototype implementation.

Key words: Identity Theft, Phishing and Social Engineering, Fraud Prevention, Secure Banking and Financial Web Services.

1 Introduction

In *phishing*, an automated form of social engineering, criminals use the Internet to fraudulently extract sensitive information from businesses and individuals, often by impersonating legitimate web sites. The potential for high rewards (e.g., through access to bank accounts and credit card numbers), the ease of sending forged email messages impersonating legitimate authorities, and the difficulty law enforcement has in pursuing the criminals has resulted in a surge of phishing attacks: estimates suggest that phishing affected 1.2 million U.S. citizens and cost businesses billions of dollars in 2004 alone [40]. Phishing also leads to additional business losses due to consumer fear. Anecdotal evidence suggests that an increasing number of people shy away from Internet commerce due to the threat of identity fraud, despite the tendency of US companies to assume the risk for fraud. Also, many users now default to distrusting any email they receive from financial institutions [16].

Current phishing attacks are still relatively modest in sophistication and have substantial room for improvement, as we discuss in Section 2.2. Thus, the research community and corporations need to make a concentrated effort to combat the increasingly severe economic consequences of phishing. Unfortunately, as we discuss in Section 8, current anti-phishing techniques do not offer adequate safeguards for ordinary users.

We present three main contributions in this paper. First, we propose several design principles needed to counter phishing attacks: 1) sidestep the arms race, 2) provide mutual authentication, 3) reduce reliance on users, 4) avoid dependence on the browser's interface, and 5) forgo network monitoring. Anti-phishing solutions that fail to follow these principles will likely be overcome or circumvented by phishers.

Second, to fulfill our design principles, we propose a foolproof anti-phishing system that does not rely on users to *always* make the correct security decision. Our mutual authentication protocol uses a trusted device (e.g., a cellphone) both to manage a second authenticator for the user and to authenticate the server. Since a user cannot readily disclose the additional authenticator to a third party, attackers must obtain the user's password *and* compromise the trusted device to gain account access. By making the trusted device an active participant in the authentication process, our protocol protects the users against Man-in-the-Middle attacks.

Our approach also defends against keyloggers and other mechanisms designed to monitor user input. The user can easily employ our scheme across multiple platforms without relying on the information in the browser's display.

Finally, we demonstrate the practicality of our system with a prototype implementation. We use a cellphone as the trusted device, and we show that the system introduces minimal overhead. In addition, the server-side changes are minor, as well as backwards compatible.

2 Problem Definition

In this section, we consider various formulations of the phishing problem and survey phishing tactics, both those in use today and those likely to appear in the near future. We also consider the aspects of user behavior typically exploited by phishing attacks.

2.1 Goals and Assumptions

In this section, we enumerate the goals of an anti-phishing technique, arranged in decreasing order of protection and generality:

1. Ensure that a user's data only goes to the intended recipient.
2. Prevent a user's data from reaching an untrustworthy recipient.
3. Prevent an attacker from abusing a user's data.
4. Prevent an attacker from modifying a user's account.
5. Prevent an attacker from viewing a user's account.

Our scheme guarantees the last two goals via technical measures. Clearly, an ideal solution would also address the first goal. However, divining a user's intentions remains a difficult problem, particularly when even the user may find it difficult to quantify his or her precise intentions. The next two goals, while more constrained than the first, require complete control over the user's data. Although we present techniques to assist with the goal of preventing the user's data from reaching an untrustworthy recipient, ultimately, we cannot guarantee this result, since a determined user can always find some means of disclosing personal information to an adversary.

To realize our goals, we assume users can be trusted to correctly identify sites at which they wish to establish accounts. We justify this assumption on the basis of the following observations. First, phishing attacks generally target users with existing accounts. In other words, the phishers attempt to fool a victim with an online account into revealing information that the phishers can use to access that account. Second, users typically exercise greater caution when establishing an account than when using the account or when responding to an urgent notice concerning the account. This results in part from the natural analogue of the real world principle of caveat emptor, where consumers are accustomed to exercising caution when selecting the merchants they wish to patronize. However, consumers in the real world are unlikely to encounter a Man-in-the-Middle attack or an imitation store front, and so they have fewer natural defenses when online. Our solution addresses these new threats enabled by the digital marketplace. Our approach is largely orthogonal to existing anti-phishing solutions based on heuristics, and it can be combined with these earlier schemes, particularly to protect the user from a phishing attack during the initial account establishment.

2.2 Attacks

A typical phishing attack begins with an email to the victim, supposedly from a reputable institution, but actually from the phisher. The text of the message commonly warns the user that a problem exists with the user's account that must immediately be corrected. The victim is led to a spoofed website designed to resemble the institution's official website. At this point, the phishing site may launch a passive or an active attack. In a passive attack, the web page prompts the victim to enter account information (e.g., username and password) and may also request other personal details, such as the victim's Social Security number, bank account numbers, ATM PINs, etc. All of this information is relayed to the phisher, who can then use it to plunder the user's accounts. In an active attack, the phisher may act as a man-in-the-middle attacker, actively relaying information from the legitimate site to the user and back.

While early phishing emails typically employed plain text and grammatically incorrect English, current attacks demonstrate increased sophistication. Phishing emails and websites often employ the same visual elements as their legitimate counterparts. As a result, spoofed sites and legitimate sites are virtually indistinguishable to users. Phishers also exploit a number of DNS tricks to further obscure the nature of the attack. The spoofed site may use a domain name like `www.ebay.com.kr`, which very closely resembles eBay's actual domain, but instead points to a site in Korea. Some attacks use obscure URL conventions to craft domain names like `www.ebay.com@192.168.0.5`, while others exploit bugs in the browser's Unicode URL parsing and display code to conceal the site's true domain name [21].

Although most phishing attacks are initiated via email, there are many other potential means of initiation. The phisher could contact the victim via Instant Messenger, via a popup or other advertisement on another website, or even via fax [22]. Phishers can also exploit mistyped URLs by registering domain names like `gooogle.com` or `goggle.com`, or even employ techniques to artificially inflate their rankings in search engines. To make matters worse, researchers have discovered automated phishing kits circulating online that enable novice phishers to employ some of these techniques [36].

Attackers have also been quick to exploit attempts at user education. For instance, many users believe that a transaction is secure if they see the 'lock' icon displayed in the browser window. One possible attack uses JavaScript to display a spoofed lock image in the appropriate location [43]. Phishers may also acquire their own SSL certificate, relying on users' inability or unwillingness to verify the certificates they install. There have also been cases in which Certificate Authorities issued certificates to attackers posing as legitimate Microsoft employees [26]. Phishers can also try to confuse users by simultaneously loading a legitimate page and a spoofed page using HTML frames or popups. Unfortunately, even these techniques barely scratch the surface of potential phishing scams.

Despite the advances and innovations discussed above, phishing attacks are continuously evolving into increasingly sophisticated forms. For example, attackers have begun targeting specific individuals within an organization. These highly customized attacks, dubbed *spear-phishing*, often try to trick employees into installing malware or revealing their organizational passwords [31, 23]. As a more general form of advanced attack, Jakobsson introduces the notion of context-aware phishing in which an attacker exploits some knowledge about the victim in order to enhance the efficacy of the attack [19]. In a user study, Jakobsson found that context-aware phishing attacks dramatically enhanced the probability of a successful attack, from 3% percent for an ordinary attack to 48-96% for a specially-crafted context-aware attack. Another attack variant uses socially-aware phishing. In a socially-aware attack, the phisher uses publicly available information to craft an email that purports to come from someone the victim knows and trusts. To defend against phishing attacks, organizations are in a constant race to detect and take down phishing sites. In the future, this could become even more difficult with *distributed phishing attacks* [20], where each page a user visits is hosted at a different location and registered to a different owner.

2.3 User Issues

In this section, we consider user-related issues for phishing. Some of these observations were also made by Dhamija and Tygar [9].

First, users exhibit certain tendencies that inherently undermine security. Security is often a secondary concern; few users start a web browser with the objective of "doing security."

Users want to make purchases, check their accounts and authorize payments online. Because of this, users will tend to ignore or, if they become too invasive, circumvent or disable security measures. Similarly, users have become habituated to ignoring strange warning boxes that appear when they access secure sites, and they blithely click through such warnings. Moreover, prior work shows that humans pick poor passwords with low entropy [42] and readily volunteer them to complete strangers [2]. Finally, users have become accustomed to computers and websites behaving erratically. They will often attribute the absence of security indicators to non-malicious errors [41]. In addition, most users cannot distinguish between actual hyperlinks and spoofed hyperlinks that display one URL but link to a different URL (i.e., URLs of the form: `<a href='http://phishing.org/'> <img src='ebay-url.jpg'> </a>`). Furthermore, users are unable to reliably parse and understand domain names or PKI certificates.

Clearly, current technology makes it difficult for even a knowledgeable user to consistently make the right decision, particularly when security is not a primary goal. As a result, we argue that anti-phishing techniques must minimize the user's security responsibilities.

3 Design Principles

Based on the previous discussion, we advocate the following set of design principles for anti-phishing tools:

Sidestep the arms race. Many anti-phishing approaches face the same problem as anti-spam solutions: incremental solutions only provoke an ongoing arms race between researchers and adversaries. This typically gives the advantage to the attackers, since researchers are permanently stuck on the defensive. As soon as researchers introduce an improvement, attackers analyze it and develop a new twist on their current attacks that allows them to evade the new defenses. Instead, we need to research fundamental approaches for preventing phishing. As Clayton noted, we need a "Kilimanjaro effect," where the level of security overwhelms potential attackers, and only the most determined (and skilled) will succeed [7].

Provide mutual authentication. Most anti-phishing techniques strive to prevent phishing attacks by providing better authentication of the server. However, phishing actually exploits authentication failures on both the client and the server side. Initially, a phishing attack exploits the user's inability to properly authenticate a server before transmitting sensitive data. However, a second authentication failure occurs when the server allows the phisher to use the captured data to login as the victim. A complete anti-phishing solution must address both of these failures: clients should have strong guarantees that they are communicating with the intended recipient, and servers should have similarly strong guarantees that the client requesting service has a legitimate claim to the accounts it attempts to access.

Reduce reliance on users. The majority of current phishing countermeasures rely on users to assist in the detection of phishing sites and make decisions as to whether to continue when a potentially phishy site is found. Unfortunately, as discussed in Section 2.3, users are in many ways unsuited to authenticating others or themselves to others. As a result, we must move towards protocols that reduce human involvement or introduce additional information that cannot readily be revealed. These mechanisms add security without relying on perfectly correct user behavior, thus bringing security to a larger audience.

Avoid dependence on the browser's interface. The majority of current anti-phishing approaches propose modifications to the browser interface. Unfortunately, the browser interface is inherently insecure and can be easily circumvented by embedded JavaScript applications that mimic the "trusted" browser elements. In fact, researchers have shown mechanisms that

The User Experience
Alice lives in New York and has an account at the National Bank of Anguilla. She often worries about the security of her online account. Recently, the bank began offering the Phoolproof cellphone authentication system to its customers. Alice is thrilled, but she cannot go to the bank in person to sign up. Alice contacts the bank. The bank mails a randomly chosen shared secret to the postal address on file.
When Alice receives the shared secret in the mail, she logs into the National Bank of Anguilla web page and navigates to the cellphone authentication signup page. The signup page prompts her to enter the shared secret into her cellphone (see Section 4.1 for technical details and alternatives). Alice confirms she wants to create a new account on her cellphone, and a bookmark for the National Bank of Anguilla then appears in her phone's list of secure sites.
From then on, whenever Alice wants to access her account, she navigates to the Anguilla bookmark on her cellphone, as shown in Figure 1. The phone directs her browser to the correct website, and Alice enters her username and password to login (see Section 4.2 for technical details). After login, the interaction with her bank remains unchanged.

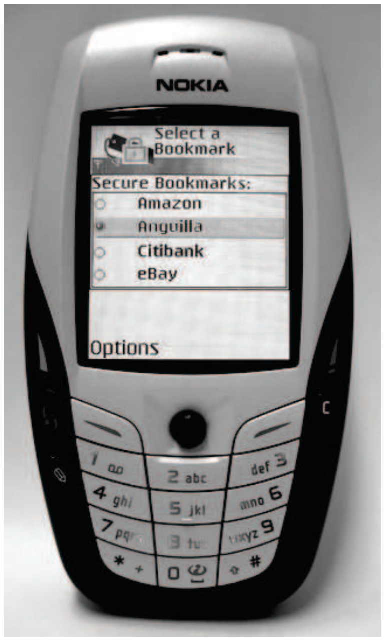

Fig. 1. Cellphone User Interface *The cellphone displays the secure bookmarks for sites at which the user has established accounts.*

imitate a secure SSL web page by forging security-related elements on the screen [43]. Even recent anti-phishing proposals that create trusted browser windows or input mechanisms are ultimately still vulnerable to JavaScript attacks [9, 33] Given the complexity of current web browsers and the multitude of attacks, we propose to avoid reliance on browser interfaces.

Forgo network monitoring. A naive approach to phishing prevention might involve monitoring a user's outgoing communication and intercepting sensitive data in transit. Unfortunately, this approach is unlikely to succeed. For example, suppose this approach is implemented to monitor information transmitted via HTML forms. An attacker could respond by using a Java applet or another form of dynamic scripting to transmit the user's response. Worse, client-side scripting could easily encrypt the outgoing data to prevent this type of monitoring entirely. In the end, this approach is unlikely to provide a satisfactory solution.

4 Our Phoolproof Anti-Phishing System

While no automated procedure can provide complete protection, our protocol guards the secrecy and integrity of a user's existing online accounts so that attacks are no more effective than pre-Internet scams (e.g., an attacker may still be able to access a user's account by subverting a company insider). We base our system on the observation that users should be authenticated using an additional authenticator that they cannot readily reveal to malicious parties. Our scheme establishes the additional authenticator on a trusted device, such that an attacker must compromise the device *and* obtain the user's password to access the user's account.

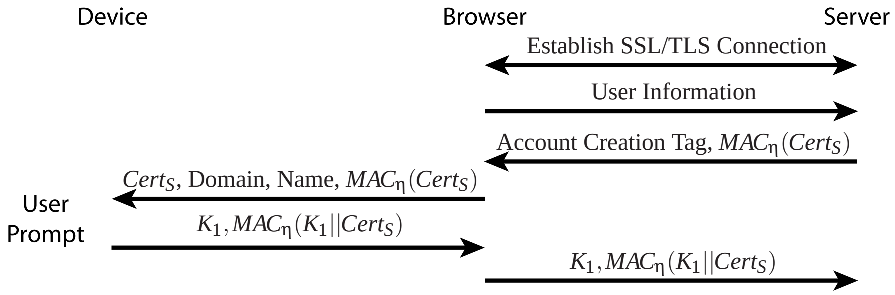
Device
Browser
Server
Establish SSL/TLS Connection
User Information
Account Creation Tag, $MAC_\eta(Cert_S)$
$Cert_S$, Domain, Name, $MAC_\eta(Cert_S)$
User Prompt
$K_1, MAC_\eta(K_1||Cert_S)$
$K_1, MAC_\eta(K_1||Cert_S)$

Fig. 2. Account Setup *Protocol steps for establishing a new user account.*

The trusted device in our system can take the form of a cellphone, PDA or even a smart watch; in this paper, we assume the use of a cellphone. Users cannot readily disclose the authenticator on the cellphone to a third party, and servers will refuse to act on instructions received from someone purporting to be a particular user without presenting the proper authenticator. As discussed in Section 8, our technique is one of the first systems to prevent active Man-in-the-Middle attacks. In addition, the use of the cellphone allows us to minimize the effect of hijacked browser windows and facilitates user convenience, since it can be used at multiple machines. We assume that the user can establish a secure connection between their cellphone and their browser and that the cellphone itself has not been compromised. We discuss these assumptions further in Section 5.2.

Below, we explain how a user creates an account (or updates an existing account) using our protocol. We then define the protocol for account usage, as well as steps for recovering if the user's trusted device is lost or compromised.

4.1 Setup

To enable our system for an online account, the user must establish a shared secret with the server. This can be done using one of the out-of-band channels suggested below. These mechanisms for establishing a shared secret rely on institutions to implement measures that ensure 1) their new customers are who they say they are, and 2) the information in existing customers' files is accurate. Institutions have dealt with this problem since well before the existence of computers, and thus, they have well-established techniques for doing so.

The out-of-band channel used for establishing a shared secret can take many forms. For example, banks often utilize the postal service as a trusted side-channel. Alternatively, a telephone call may suffice. Banks could provide the shared secret at ATMs by displaying the shared secret in the form of a barcode that the user could photograph with the camera on a cellphone [25, 32]. As another possibility, initial account setup could be performed on the premises of the financial institution. That way, employees can be trained to assist users with setup; users' identification can be checked in person; and users can trust that they are associating with the correct institution. Trusted financial institutions could also provide setup services for organizations that lack brick-and-mortar infrastructure, such as online vendors.

Using one of the mechanisms discussed above, the institution sends a randomly chosen secret η to the user. The secret should be of sufficient length (e.g., 80-128 bits) to prevent brute-force attacks. The user navigates to the institution's website and initiates setup. The setup steps are summarized in Figure 2 and described below. The server responds with a specially crafted HTML tag (e.g., `<!-- SECURE-SETUP -->`), which signals the browser that account setup has been initiated. The server also authenticates its SSL/TLS certificate by including a MAC of the certificate, using the shared secret η as a key.

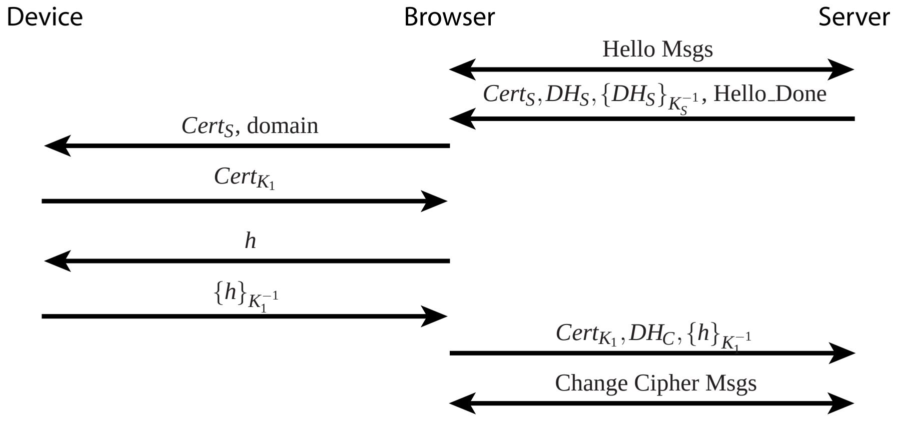

Fig. 3. Secure Connection Establishment *The browser establishes an SSL/TLS connection to the server using client authentication, with help from the cellphone. DH_S and DH_C represent the Diffie-Hellman key material for the server and client respectively, and h is a secure MAC of the handshake messages. See Section 4.2 for additional details.*

The browser contacts the cellphone via Bluetooth,[1] transmitting the server's SSL/TLS certificate, domain name, site name and MAC to the phone. The cellphone prompts the user to confirm the account creation (to avoid stealth installs by malicious sites) and enter the shared secret provided by the institution (if it has not already been entered, e.g., at the ATM or at the financial institution). It also verifies the MAC on the server's certificate and aborts the protocol if the verification fails. Assuming verification succeeds, the cellphone creates a public/private key pair $\{K_1, K_1^{-1}\}$ and saves a record associating the key pair with the server's certificate. It also creates a *secure bookmark* entry for the site, using the site's name and domain name. The cellphone sends the new public key authenticated with a MAC, using the shared secret as a key, to the server. The server associates the public key with the user's account, and henceforward, the client must use the protocol described in the next section to access the online account. All other online attempts to access the account will be denied.[2]

4.2 Secure Connection Establishment

Once the user's account has been enabled, the server will refuse access to the account unless the user is properly authenticated via the established public key pair *and* username/password combination. Thus, even if the user is tricked into revealing private information to a phisher or a social engineer, the attacker still cannot access the user's account.

A user who wishes to access the account must always initiate the connection using the secure bookmark on the cellphone. As an alternative, we could have the cellphone detect when a user navigates to a previously registered site. However, a cellphone is ill-equipped to detect if the user visits a phishing site and thus will be unable to prevent the user from disclosing private information to malicious parties. While a phisher would still be unable to access the user's account (without compromising the cellphone), we prefer to help prevent this unnecessary disclosure (see Section 5 for additional discussion).

When the user selects a secure bookmark on the cellphone, the cellphone directs the browser to the associated URL. The use of secure bookmarks provides the user with a higher

[1] Our system is not exclusive to Bluetooth. Any mechanism that allows the user's trusted device to communicate with the browser (e.g., infrared, 802.11, USB cable, etc.) will suffice.

[2] Note that this does not preclude Alice from conducting business in person, for example.

degree of server authentication and helps to protect the user from inadvertently arriving at a phishing site, either via a spoofed or a mistyped URL. When the remote server provides its SSL/TLS certificate, the browser forwards the certificate to the cellphone. If the certificate does not match the certificate previously provided, the cellphone closes the browser window and displays a warning message. If a server updates its certificate, then we need a protocol to update the server certificate stored on the cellphone; for example, the server could send the new certificate along with a signature using the previous private key, and upon successful verification, the cellphone can update the certificate it has stored.

If the certificate check is successful, the browser and the server then establish an SSL/TLS connection [11, 14]. The cellphone assists the browser in performing the client authentication portion of the SSL/TLS establishment, using the public key pair associated with this site (the SSL/TLS protocol includes a provision for user authentication, but this is rarely used today). Figure 3 summarizes the messages exchanged. Essentially, the browser initiates an SSL/TLS connection with Ephemeral Diffie-Hellman key agreement. After agreeing on the cryptographic parameters in the Hello messages, the server sends:

$$Cert_S, g, p, g^s mod\, p, \{g, p, g^s mod\, p\}_{K_S^{-1}} \tag{1}$$

(i.e., its certificate, its ephemeral Diffie-Hellman key information and a signature on the key information) to the client. The browser retrieves the appropriate user certificate $Cert_{K_1}$ from the cellphone based on the server's certificate and domain. Then, the browser generates the necessary Diffie-Hellman key material and calculates a secure hash of the SSL/TLS master secret $\mathcal{K}$ (which is based on the derived Diffie-Hellman key) and all of the previous handshake messages (as well as the client's choice of Diffie-Hellman key material), HM, as follows:

$$h = MD5(\mathcal{K}||pad2||MD5(HM||\mathcal{K}||pad1))||\text{SHA-1}(\mathcal{K}||pad2||MD5(\text{SHA-1}||\mathcal{K}||pad1)) \tag{2}$$

(where $||$ represents concatenation) and sends the hash to the cellphone. The cellphone replies with a signature on h. Note that as long as the phone remains uncompromised, an attacker cannot produce this signature, and hence cannot successfully authenticate as the user. The browser forwards the signature to the server, along with the user's certificate and the client's Diffie-Hellman key material:

$$Cert_{K_1}, g^c mod\, p, \{h\}_{K_1^{-1}} \tag{3}$$

The browser and the server then exchange the final phase of an SSL/TLS negotiation. Once the user has been authenticated and the SSL/TLS connection has been established, the user can use the browser to conduct transactions and account inquiries as usual. Note that we do not change the SSL/TLS protocol; we merely use the cellphone to assist the browser establish a session key with the server.

4.3 Recovery

Inevitably, users will lose or break their cellphones, or replace them with newer models. When this happens, the user must revoke the old keys and establish a new key pair with a new cellphone. In the case of a lost cellphone, revocation prevents an attacker from accessing the user's accounts.

To revoke the old key pairs, we favor using a process that exists today: the user calls the institution via telephone. This is a well-established, familiar process. Today, customers already call credit card companies to report the loss of a card and to freeze any transactions on the account. With the loss of a cellphone, users would still call the institutions to revoke their keys. The institution would then send the information needed to establish a new key pair using the techniques described in Section 4.1.

We initially considered other methods, such as storing revocation information in the user's browser or on a USB key. However, telephone calls are superior for three reasons. First, users already know how to call customer service. The reuse of an existing business process reduces the costs – mental and monetary – for all parties. Second, cellphones are mobile devices that travel with their users, and users may lose them anywhere. A user whose cellphone is lost on a business trip should act immediately to minimize financial (or other) losses; waiting to access the revocation information stored at home is not acceptable. Finally, since revocation information is rarely used, it is easily lost. For example, if revocation information is stored on paper, CD's, or USB keys, it can be misplaced or damaged.

5 Security Analysis

In this section, we discuss the effectiveness of various attacks on our system.

5.1 Hijacking Account Setup or Re-establishment

The largest vulnerability in our system arises during account setup (or re-establishment), when the user must ensure that the account is created at a legitimate site. The server also faces an authentication problem, since it must ensure that the person creating the account is the person described by the user information submitted. However, as we discuss in Section 4.1, this threat can be mitigated by having users establish accounts in person, at trusted businesses.

A clever phisher may have a user's key pair revoked and hijack the account when the user tries to re-establish a new key pair. We do not consider this to be a grave threat, however. Phishing is so successful because the perpetrators target large numbers of people with a low success rate. It would be difficult for phishers to target a large number of people without attracting attention; a bank would surely notice if thousands of customers suddenly needed new key pairs. In addition, phishers typically have limited resources in the physical world. Sending information for key re-establishment through postal mail or requiring users to verify their identity in person would greatly reduce the effectiveness of a phishing attack.

5.2 Theft of the Trusted Device

Since the user's cellphone (or PDA) holds cryptographic keys for all of the user's accounts, device theft is a risk. After stealing the device, an attacker would still need the user's password(s) in order to compromise the accounts (a problem sufficient to deter casual attackers). The attacker must obtain the passwords before the user discovers the theft and revokes the stored keys. Nonetheless, additional layers of security may be desirable. For example, the cellphone could require the user to enter a PIN number or use biometrics to authorize use of the keys. A more security-conscious user could consider a tamper-resistant storage module for the cellphone to reduce the possibility of leaking the secret keys.

5.3 Malware on the Trusted Device

With the advent of more powerful cellphones and network-enabled PDAs, malware on mobile devices will become an increasingly serious problem. Attacks are inevitable – particularly if mobile devices are used to protect financial accounts.

Numerous vendors have released anti-malware software for mobile devices. More high-profile attacks may be required before the software becomes ubiquitous on mobile devices, as it is on computers.

For additional security, we could leverage a Trusted Platform Module (TPM) that will likely exist on future cellphone architectures. The keys would reside in the TPM's trusted storage facility. In the absence of additional security hardware, we could instead use recent advances in software attestation [35] to verify the integrity of both the trusted device and the user's computer. When the user's cellphone contacts the computer, each device attests to the security of its current state, and the SSL/TLS connection only proceeds if both parties are satisfied. As a result, a successful attack would require simultaneous compromise of both the user's cellphone and computer.

In addition, we can leverage the capture-resilient cryptographic mechanisms proposed by MacKenzie and Reiter [24]. In their approach, secrets and cryptographic operations are split up and performed on the mobile device and a server. Compromising either the mobile device or the server reveals no useful information. After loss of the mobile device, the user can revoke the information stored on the server.

5.4 Malware on the Computer

Our system protects the user against many forms of malware. For example, standard key-loggers would be ineffective, since they can only capture user input – and not the private key stored on the cellphone. However, without additional resources, we cannot protect users against certain classes of malicious code installed on users' computers. The two largest threats are malicious modifications to the browser and operating system kernel compromises. If the user's browser has been compromised, then the malicious code could use the cellphone to login to a legitimate site and subsequently redirect the connection to a phishing site. A kernel compromise would allow a similar attack. Both attacks require a significant and sophisticated compromise of the user's computer. As mentioned earlier, we can use new security hardware such as the TPM or software attestation techniques to mitigate these threats.

5.5 Attacks on the Network

Possible network-based attacks include Man-in-the-Middle attacks, pharming[3] attacks, and domain hijacking. None of these attacks will succeed against our system. By storing the user's public key, the server prevents a Man-in-the-Middle attack, since the attacker will not be able to attack the authenticated Diffie-Hellman values from the ephemeral Diffie-Hellman exchange. By checking the certificate provided by the server against the stored certificate, the cellphone even protects the user from DNS poisoning and domain hijacking. Thus, our scheme provides very strong guarantees of authenticity to both the client and the server and stops virtually all forms of phishing, DNS spoofing and pharming attacks.

5.6 Local Attacks on Bluetooth

Phishing attacks rely on the attacker's ability to target a large number of users and swindle them quickly without being caught. As a result, phishing attacks are typically conducted remotely. To provide an additional layer of protection, we can use existing research (e.g., from McCune et al. [25]) to establish a secure connection between the user's device and the computer they use, preventing attacks on the Bluetooth channel.

6 Discussion

6.1 Infrastructure

As described above, our protocol requires very minimal changes to existing infrastructure. In Section 7, we provide specific details of our prototype implementation to demonstrate the limited changes necessary to support our protocol.

[3] Pharming attacks exploit vulnerabilities in a DNS server to redirect users to another site. Such DNS attacks are powerful in conjunction with phishing, since the domain name appears to be correct.

For servers, the primary change is the addition of an extra record associated with a user's account to store the user's public key. Servers must also respond to account creation requests by adding an extra HTML tag[4] to the page. The authentication of the user's public key uses existing options in the SSL/TLS protocol, so that the SSL/TLS protocol remains unchanged and client authentication code requires only minor tweaks.

From the client's perspective, the browser's portion of the protocol can be implemented as a simple browser extension, as we have done in our prototype, or it could eventually be incorporated into the browser itself.

As for the user's trusted device, cellphones today are one of the first examples of truly ubiquitous computing. According to a survey in 2003, over 49% of Americans and 90% of Europeans have cellphones [37]. However, as mentioned earlier, our protocol can work just as well with a user's PDA or other mobile-computing device (e.g., a smart watch). Using Bluetooth as a basis for communication between the trusted device and the computer is also increasing practical. According to the Bluetooth SIG, over five million units are shipped with Bluetooth every week, with an installed base of over 500 million units at the end of 2005 [4].

As we discuss in Section 7, the software for the user's trusted device can be developed in Java, simplifying portability between devices.

6.2 Deployment Incentives

Our system provides strong deployment incentives for both parties involved in online transactions. Consumers will be motivated to use the system, since it imparts very strong guarantees regarding the integrity of their online accounts and helps prevent them from inadvertently visiting a phishing website. Financial institutions and merchants will want to adopt a system that will help reduce losses due to phishing attacks. Our scheme can be deployed by individual organizations without the need for universal adoption and deployment. Each server that deploys the system benefits, regardless of whether or not it is adopted by other sites. In addition, the scheme can be deployed alongside legacy authentication so that legacy users will still be able to access their accounts.

6.3 Convenience

There are many other two-factor authentication schemes using stand-alone accessories, such as security tokens or smart cards [34]. However, each organization must currently issue its own accessory, and users are saddled with multiple accessories that may be confused or lost. Our system enables all this functionality to be consolidated onto one device the user already carries. Furthermore, our approach prevents Man-in-the-Middle attacks, which are still possible with many of the accessories since a one-time password entered into a browser window can be captured by a phishing site and exploited to hijack the user's account. Moreover, browser-based countermeasures can be inconvenient, since state kept on the browser may not be easily portable.

7 Prototype Implementation

To evaluate the usability and performance of our scheme, we developed a prototype on a cellphone, a web browser and a server. We discuss the details and performance results below.

[4] The tag is designed so that legacy clients will simply ignore it.

7.1 Implementation Details

Equipping a server with our system required very minimal changes, namely changes to two configuration options and the addition of two simple Perl scripts. From the server's perspective, our scheme requires no changes to the SSL/TLS protocol. Indeed, most major web servers, including Apache-SSL, Apache+mod_ssl and Microsoft's IIS already include an option for performing client authentication. In our case, we used Apache-SSL and enabled the `SSLVerifyClient` option that indicates that clients may present certificates, but the certificates need not be signed by a trusted Certificate Authority (since our client certificates are self-signed). We also enabled the `SSLExportClientCertificates` option that exports information about the client's certificate to CGI-accessible variables. Aside from these two minor configuration changes, we only needed two additional CGI scripts (written in Perl) to implement the server's side of the protocol. One script handles account creation and writes user information and public keys to a file. When the client attempts to use the account, it provides a self-signed certificate as part of the normal SSL/TLS authentication process. The server's existing SSL/TLS module verifies that the signature in the certificate corresponds to the public key enclosed and provides the information in the client's certificate to the authentication script. The authentication script checks the public key in the certificate against that associated with the user's account. If the keys match, then the authentication script permits the client to access the site. This approach has several benefits. First, the changes required are extremely minor and nonintrusive. Second, it still allows legacy clients to establish an SSL/TLS connection with the server. The authentication script can then detect whether the client has presented a legitimate certificate. If the script detects a legacy client, it can make a policy decision as to whether to allow the client access to the account, allow restricted access to the account, or redirect the client to the account creation page.

On the client side, we developed an extension to Firefox, an open-source web browser, to detect account creation. When the extension detects a page containing the account creation tag, it signals the cellphone with the appropriate information, and passes the cellphone's reply to the server. Similarly, when the user selects a secure bookmark on the cellphone, the cellphone sends the URL to the extension, which redirects the browser to the appropriate site. We also chose to apply a small patch to the Firefox code that handles the client authentication portion of the SSL/TLS exchange.[5] The patch passes the server's certificate to the cellphone, along with a hash of the SSL/TLS handshake messages and receives from the cellphone a certificate for the user's public key and a signature on the hash. The browser can then use these items to complete the SSL/TLS handshake. By involving the cellphone in the SSL/TLS computations, we guarantee that the private key for the account never leaves the phone, preventing even a compromised browser or OS from accessing it.

Our prototype runs on a Nokia 6630 cellphone. We developed a Java MIDlet (an application conforming to the Mobile Information Device Profile (MIDP) standard) that provides the functionality described in Section 4 with a user-friendly interface. A Java implementation also simplifies porting the code to other devices. For the cryptographic operations, we used the light-weight cryptography library provided by Bouncy Castle [38]. Since key generation can require a minute or two, we precompute keys when the user first starts the application, rather than waiting until an account has been created. When the cellphone receives an account creation packet from the browser extension, it selects an unused key pair, assigns it to the server information provided by the browser extension, and then sends the key pair and the appropriate revocation messages to the browser extension. When the user selects a secure bookmark (see Figure 1), the cellphone sends the appropriate address to the browser extension. It also computes the appropriate signatures during the SSL/TLS exchange.

[5] Instead of patching Firefox, we could also implement our scheme as an SSL/TLS proxy on the user's computer. This would enable our solution to work with proprietary browsers as well. However, the patch to Firefox was small and straightforward, so we chose that route for testing purposes.

	Time (s)	[Min, Max] (s)
Key Creation	75.0	[29.8, 168.3]
Account Creation	0.4	[0.3, 0.5]
Site Navigation	0.2	[0.1, 0.2]
SSL/TLS Assistance	1.7	[1.6, 1.9]

Table 1. *This table summarizes the performance overhead imposed by our scheme. The averages are calculated over 20 trials, and the keys created are 1024-bit RSA key pairs. Note that key creation happens offline and thus has little or no impact on the user's experience.*

7.2 Performance

If our system is to provide a realistic defense against phishing attacks, it must impose minimal overhead, since a solution that significantly slows the web browsing experience will be unlikely to be adopted. Table 1 summarizes our performance measurements. These results represent the average over 20 trials, each run on a cold cellphone cache. Clearly, key creation takes the largest amount of time (which is understandable, given that the cellphone must create a 1024-bit RSA key pair), but since we precompute the keys, the user will not be affected by this overhead. We could also make use of the efficient key generation technique proposed by Modadugu et al. [27] to significantly decrease the delay. More importantly, account creation time is negligible, as is the delay for the cellphone to direct the browser to a given domain. The overhead for using the system during an SSL/TLS exchange requires less than two seconds on average, which is tolerable in most cases. Furthermore, newer phones already promise better performance, and an optimized implementation of the necessary cryptographic primitives in C would likely reduce the overhead by an order of magnitude (our current RSA implementation is written entirely in Java), though at the potential cost of additional overhead when porting the code to a new device. Together, these improvements would reduce the usage overhead to well under a second.

8 Related Work

The importance of the phishing problem has attracted much academic and industrial research. Many of the systems described below represent complementary approaches and could be used in conjunction with our system, particularly to help protect the user during account setup. We discuss related work in three categories: heuristic approaches, password modification, and origin authentication.

8.1 Heuristics

A popular initial approach for preventing phishing attempts is to find a pattern in phishing web sites and then alert the user if a given site matches the pattern. Several browser toolbars have been proposed to perform this function, for example SpoofGuard [5], TrustBar [18], eBay Toolbar [12], and SpoofStick [8]. Among other heuristics, these toolbars detect malicious URLs and inform the user about the true domain of the site visited. The Net Trust system incorporates information from users' social networks, as well as centralized authorities, to help users make decisions about a website's trustworthiness [15]. Unfortunately, heuristics are inherently imprecise and invite attackers to adapt to the defenses until they can bypass the heuristics. Such an approach can lead to an arms race as we describe in Section 3, with all of the problems it entails. In addition, Wu et al. found that 13-54% of users would still visit a phishing website, despite warnings from an anti-phishing toolbar [41].

8.2 Modified Passwords

Phishers often exploit the tendency of users to pick weak passwords and to re-use the same passwords at several websites. If a phisher obtains a password at a low-security site, they can use it to login to a high-security site as well.

One-time passwords are widely used in several contexts, including the S/Key system [17] and corporate uses such as Citibank [6]. The RSA SecurID system is a time-based one-time password, where the password is generated on a hardware token [34]. The user must enter the code in a web form and submit it to the server to show that she possesses the trusted device, but there is no server authentication on the user's part. In addition, the system is vulnerable to an active Man-in-the-Middle attack, since a phisher can intercept the value from the user and then use it to access the user's account. The PwdHash approach uses a cryptographic hash function computed on the user's password and the site name to derive a unique password for each site [33]. PwdHash is a promising system, but is ineffective against pharming or DNS spoofing attacks where a phisher presents the correct domain name to the browser but redirects the request to its server. In the case of DNS attacks, PwdHash will hand the correct password for the site to the phisher. Moreover, PwdHash does not prevent a phisher from breaking a weak master password using dictionary attacks.

Another approach is single-sign-on, where users sign in to a single site that will subsequently handle all authentications with other sites, but so far such systems have encountered consumer resistance, since they involve storing sensitive user data with a third party. If these services did grow in popularity, they would undoubtedly attract the same attention from phishers currently visited on individual sites. Another approach is "Verified by VISA," where merchants redirect clients to a special VISA site which requires a username and password to authenticate the transaction [1].

Unfortunately, none of these approaches provide sufficient protection against Man-in-the-Middle attacks, particularly if the phisher also uses DNS spoofing. As the user enters personal information into the phishing website, the phisher can forward the information to the legitimate banking site. Once authenticated, the adversary has full control over the hijacked connection. Banks have already reported such attacks against their one-time password systems [29]. Our approach precludes such Man-in-the-Middle attacks because the cell phone and server mutually authenticate each other and establish a session key end-to-end.

8.3 Origin Authentication

In this class of countermeasures, researchers propose user-based mechanisms to authenticate the server. Ideally, if the user arrives at a malicious website, he or she will detect that the phishing site is not the correct web site.

Jakobsson presents a theoretical framework for phishing attacks [19]. He also proposes better email authentication to prevent phishing email, in addition to better secrecy protection for user email addresses (such that phishers have a harder time harvesting email addresses from, for example, eBay).

The Petname project [39] associates a user-assigned nickname with each website visited. If the browser loads a page from a spoofed web site, the nickname will be missing or wrong – the approach relies on users to notice either case. In addition, users will likely choose predictable nicknames (e.g., nicknaming Amazon.com's website "Amazon"), making nicknames easy to spoof.

Dhamija and Tygar propose Dynamic Security Skins (DSS) to enable a user to authenticate the server [10, 9]. In their system, a server opens a user-customized popup window that displays an image only the correct server can produce. Similar to the Petname project, this approach relies on the user to perform the verification.

Myers proposes that servers display a series of images as users type their passwords [28]. It would be difficult for phishing sites to guess the correct sequence of images, and users know what images to expect. Again, this scheme relies on the user to perform the verification.

Similarly, PassMark stores a secure cookie on the client and sets up an image associated with the account that the user should remember [30]. Unfortunately, PassMark is a proprietary system – they do not disclose a detailed description of their approach.

All of these approaches require user diligence – even a single mistake on the user's part will result in a compromised account. Several of these approaches are also susceptible to Man-in-the-Middle attacks since a phisher can simply forward information between the browser and the legitimate site.

9 Conclusion

Phishing is a significant and growing problem which threatens to impose increasing monetary losses on businesses and to shatter consumer confidence in e-commerce. We observe that phishing attacks have the potential to become much more sophisticated, making user-based protection mechanisms fragile given the user population of non-experts. Instead of relying on users to protect themselves against phishing attacks (as previous work suggests), we propose mechanisms that do not rely on the user, but are based on cryptographic operations on a trusted mobile device that many users already possess. We anticipate that our approach would be deployed for websites requiring a high level of security, and that it would ultimately help in regaining consumer confidence in using web-based commerce. In conclusion, our system satisfies the guidelines published by the FDIC, which strongly urge financial institutions to adopt two-factor authentication for Internet-based financial services by the end of 2006 [13].

Acknowledgements

The authors would like to thank Jon McCune, Jason Rouse and Ai Qi Chua for their assistance in implementing the prototype. Ari Juels, Dawn Song, and Chris Karlof provided interesting discussions and insightful comments. We would also like to thank the anonymous reviewers for their helpful suggestions.

References

1. Verified by VISA. `http://usa.visa.com/personal/security/vbv/how_it_works.html`.
2. A. Adams and M. A. Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, Dec. 1999.
3. Anti-Phishing Working Group. Phishing activity trends report. `http://antiphishing.org/apwg_phishing_activity_report_august_05.pdf`, Aug. 2005.
4. Bluetooth SIG. Bluetooth Technology Benefits. `http://www.bluetooth.com/Bluetooth/Learn/Benefits/`.
5. N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell. Client-side defense against web-based identity theft. In *NDSS*, Feb. 2004.
6. CitiBank. Virtual account numbers. `http://www.citibank.com/us/cards/tour/cb/shp_van.htm`.
7. R. Clayton. Who'd phish from the summit of kilimanjaro? In *Financial Cryptography*, pages 91–92, 2005.
8. Core Street. Spoofstick. `http://www.corestreet.com/spoofstick/`.
9. R. Dhamija and J. D. Tygar. The battle against phishing: Dynamic security skins. In *ACM Symposium on Usable Security and Privacy (SOUPS '05)*, July 2005.
10. R. Dhamija and J. D. Tygar. Phish and HIPs: Human interactive proofs to detect phishing attacks. In *Human Interactive Proofs: Second International Workshop (HIP 2005)*, May 2005.
11. T. Dierks and C. Allen. The TLS protocol version 1.0. Internet Request for Comment RFC 2246, Internet Engineering Task Force, Jan. 1999. Proposed Standard.
12. eBay. eBay toolbar. `http://pages.ebay.com/ebay_toolbar`.

13. FDIC. Authentication in an internet banking environment. Technical Report FIL-103-2005, Federal Deposit Insurance Corporation, Oct. 2005.
14. A. Freier, P. Kariton, and P. Kocher. The SSL protocol: Version 3.0. Internet draft, Netscape Communications, 1996.
15. A. Genkina, A. Friedman, and J. Camp. Net trust. Trustworthy Interfaces for Passwords and Personal Information (TIPPI) Workshop, June 2005.
16. G. Goth. Phishing attacks rising, but dollar losses down. *IEEE Security and Privacy*, 3(1):8, January–February 2005.
17. N. Haller. The S/Key one-time password system. In *Proceedings of the Symposium on Network and Distributed Systems Security*, pages 151–157, Feb. 1994.
18. A. Herzberg and A. Gbara. Trustbar: Protecting (even naive) web users from spoofing and phishing attacks. Cryptology ePrint Archive, Report 2004/155, 2004.
19. M. Jakobsson. Modeling and preventing phishing attacks. In *Financial Cryptography*, 2005.
20. M. Jakobsson and A. Young. Distributed phishing attacks. Workshop on Resilient Financial Information Systems, Mar. 2005.
21. E. Johanson. The state of homograph attacks. `http://www.shmoo.com/idn/homograph.txt`, Feb. 2005.
22. J. Leyden. Fax-back phishing scam targets paypal. `http://www.channelregister.co.uk/2005/08/11/fax-back_phishing_scam/`.
23. J. Leyden. Spear phishers launch targeted attacks. `http://www.theregister.co.uk/2005/08/02/ibm_malware_report/`, Aug. 2005.
24. P. MacKenzie and M. K. Reiter. Networked cryptographic devices resilient to capture. *International Journal of Information Security*, 2(1):1–20, Nov. 2003.
25. 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.
26. Microsoft. Erroneous VeriSign-issued digital certificates pose spoofing hazard. `http://www.microsoft.com/technet/security/bulletin/MS01-017.mspx`, 2001.
27. N. Modadugu, D. Boneh, and M. Kim. Generating RSA keys on a handheld using an untrusted server. In *RSA Conference 2000*, Jan. 2000.
28. S. Myers. Delayed password disclosure. Trustworthy Interfaces for Passwords and Personal Information (TIPPI) Workshop, June 2005.
29. Out-law.com. Phishing attack targets one-time passwords. `http://www.theregister.co.uk/2005/10/12/outlaw_phishing/`, Oct. 2005.
30. Passmark Security. Protecting your customers from phishing attacks: an introduction to passmarks. `http://www.passmarksecurity.com/`, 2005.
31. P. F. Roberts. Spear phishing attack targets credit unions. `http://www.eweek.com/article2/0,1895,1902896,00.asp`, Dec. 2005.
32. M. Rohs and B. Gfeller. Using camera-equipped mobile phones for interacting with real-world objects. In *Proceedings of Advances in Pervasive Computing*, pages 265–271, Apr. 2004.
33. B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In *14th USENIX Security Symposium*, Aug. 2005.
34. RSA Security. Protecting against phishing by implementing strong two-factor authentication. `https://www.rsasecurity.com/products /securid/whitepapers/PHISH_WP_0904.pdf`, 2004.
35. A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla. Pioneer: Verifying integrity and guaranteeing execution of code on legacy platforms. In *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, pages 1–16, Oct. 2005.
36. Sophos. Do-it-yourself phishing kits found on the internet, reveals sophos. `http://www.sophos.com/spaminfo/articles/diyphishing.html`.
37. D. Standish. Telephonic youth. `http://www.techcentralstation.com/090903C.html`.
38. The Legion of the Bouncy Castle. Bouncy Castle crypto APIs. `http://www.bouncycastle.org`.
39. Waterken Inc. Petname tool. `http://www.waterken.com/user/PetnameTool/`, 2005.
40. Wikipedia. Phishing. `http://en.wikipedia.org/wiki/Phishing`.
41. M. Wu, S. Garfinkel, and R. Miller. Users are not dependable - how to make security indicators to better protect them. Talk presented at the Workshop for Trustworthy Interfaces for Passwords and Personal Information, June 2005.
42. J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password memorability and security: Empirical results. *IEEE Security and Privacy*, 2(5):25–31, September–October 2004.
43. E. Ye and S. Smith. Trusted paths for browsers. In *Proceedings of the 11th USENIX Security Symposium*. USENIX, Aug. 2002.