Enhancing Passwords Security Using Deceptive Covert Communication

Mohammed H. Almeshekah, Mikhail J. Atallah, Eugene H. Spafford

Enhancing Passwords Security using Deceptive Covert Communication

Mohammed H. Almeshekah, Mikhail J. Atallah, and Eugene H. Spafford

Computer Science Department and CERIAS, Purdue University,
305 N. University St.,West Lafayette, IN 47907
{malmeshe,spaf,matallah}@purdue.edu
https://www.cs.purdue.edu/

Abstract. The use of deception to enhance security has shown promising results as a defensive technique. In this paper we present an authentication scheme that better protects users' passwords than in currently deployed password-based schemes, without taxing the users' memory or damaging the user-friendliness of the login process. Our scheme maintains comparability with traditional password-based authentication, without any additional storage requirements, giving service providers the ability to selectively enroll users and fall-back to traditional methods if needed. The scheme utilizes the ubiquity of smartphones; however, unlike previous proposals it does not require registration or connectivity of the phones used. In addition, no long-term secrets are stored in any user's phone, mitigating the consequences of losing it. Our design significantly increases the difficulty of launching a phishing attack by automating the decisions of whether a website should be trusted and introducing additional risk at the adversary side of being detected and deceived. In addition, the scheme is resilient against Man-in-the-Browser (MitB) attacks and compromised client machines. We also introduce a covert communication mechanism between the user's client and the service provider. This can be used to covertly and securely communicate the user's context that comes with the use of this mechanism. The scheme also incorporates the use of deception that makes it possible to dismantle a large-scale attack infrastructure before it succeeds. As an added feature, the scheme gives service providers the ability to have full-transaction authentication.

With the use of our scheme, passwords are no longer communicated in plaintext format to the server, adding another layer of protection when secure channels of communication are compromised. Moreover, it gives service providers the ability to deploy risk-based authentication. It introduces the ability to make dynamic multi-level access decisions requiring extra authentication steps when needed. Finally, the scheme's covert channel mechanisms give servers the ability to utilize a user's context information — detecting the use of untrusted networks or whether the login was based on a solicitation email.

Keywords: Authentication, Smartphone, Deception, Covert Channel

1 Introduction

A recent American Banking Association (ABA) report identified internet banking as the preferred method for customer banking [**?**]: 62% of customers named online banking as their preferred banking method, a substantial rise from 36% in 2010. At the same time, phishing has been an increasing threat — rising at an alarming rate despite all the security mechanisms banks have in place. Criminals have been stealing money by means of exploiting the ubiquity of online banking. It is estimated that the Zeus trojan alone resulted in $70 million dollars stolen from bank accounts [**?**]. Many of the currently deployed two factor authentication schemes by banks remain vulnerable to a number of attacks [**?**]. Zeus managed to bypass two factor authentication schemes employed by banks [**?**]. Adham et al. presented a prototype of a browser add-ons that, even with two factor authentication, can successfully manipulate banking transactions on-the-fly [**?**]. There is clearly a need to improve the currently deployed schemes and address their shortcomings.

Deployed schemes need to tackle the issues of stolen credentials and phishing, and to mitigate man-in-middle (MitM) and man-in-the-browser (MitB) attacks – we discuss these two attacks in the next section. In this paper we present a mechanism that address these challenges. The scheme has the following desirable characteristics:

1. It automates the decision whether a website should be trusted before the user submits her password. This enhances the ability to detect, and further deceive, an adversary launching a phishing attack by increasing the risk of conducting such attacks.
2. Resilience against the common case of using an untrusted computer and/or network for a login session (e.g., at a hotel lobby or using a guest-account when visiting another organization).
3. A hidden/covert channel facility to convey information to the server about the current authentication context. This can be utilized by the user herself and/or her client – without user involvement. Users can convey their status or doubts they harbor about the trustworthiness of the computer or network they are using for the login session. Users often know that an activity is hazardous yet engage in it nevertheless because of perceived necessity (they need to check their account balance, even in unsafe circumstances). The bank can use this user-conveyed information to grant limited access (e.g., reading account balances and paying utility bills but not carrying out transfers to other bank accounts).
4. Unlike previous schemes, our use of smartphones does not necessitate storing any permanent information on the phone, does not require the phone to have network connectivity or ability to communicate with the computer, and does not require the phone to be registered. It merely uses the smartphone's computing capability.
5. The user-friendly covert channel facility facilitates the use of honeyaccounts through which service providers can learn about the attackers, their methods of operation, which other banks and laundering accounts they use, etc.

Throughout the paper we are using the notion of a bank generically, for two reasons. First, banking is one of the prominent use-cases necessitating the use of secure authentication. Second, banks are quickly becoming the target of choice for malfeasance by evildoers. The scheme we propose is, of course, more generally applicable.

2 Background

2.1 Authentication Schemes

We are concerned with two general classifications of attacks against client-server communication: man-in-the-middle (MitM) and man-in-the-browser (MitB) as shown in figure **??**. In the former case, the adversary places herself in the communication channel between the user's computer and the server. End-to-end encryption schemes, such as SSL/TLS and IPSec, are intended to address this so that the adversary cannot observe or alter the data in the communication channel. Attackers overcome this protection by forcing the user to have an end-to-end encrypted channel with them instead of the real server, which is the case in phishing attacks. In the latter case, MitB, the attacker places herself between the user and his computer by altering the interface (browser) and manipulates the information displayed to the user in real-time. In this case even if the user employs an end-to-end encryption scheme, such as SSL/TLS, the attacker accesses the information when it is decrypted and can actively modify it before it is shown to the user.

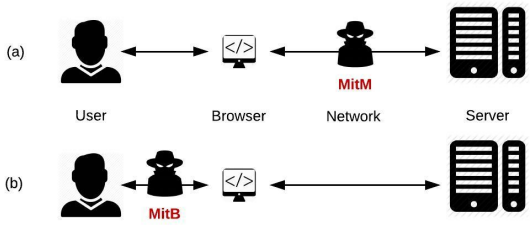

Fig. 1. Man-in-the-Middle (MitM) vs. Man-in-the-Browser (MitB)

Adham et al. identified three main authentication schemes built on the traditional username and password in the area of online banking [**?**]. These schemes are one-time password (OTP), partial transaction authentication, and full transaction authentication. They have shown that OTP schemes such as HMAC-Based One-time Password (HOTP) [**?**] or Time-based One-time Password (TOTP) [**?**] are not secure against active man-in-the-middle attacks (MitM) or man-in-the-browser (MitB) attacks [**?**]. The former can be orchestrated using

an active phishing attack, in which the adversary immediately uses the stolen credentials to impersonate the user to the bank, while the latter can be seen, as an example, in the Zeus trojan.

To address the problem of active MitB attacks, banks started to use transaction authentication [**?**,**?**]. The Chip Authentication Program (CAP) introduced by many banks requires a piece of dedicated hardware, and its protocol has a number of vulnerabilities [**?**]. A number of these hardware devices degrade the full transaction authentication to authenticate only part of the transaction, as a consequence of usability challenges [**?**]. CrontoSign [**?**] is a smartphone-based full-transaction authentication that uses the smartphone to verify the information. The scheme requires a new phone registration process that stores information on the phone, which makes the user vulnerable if her phone is compromised or stolen. In addition, it ties the user to a specific phone, hindering the usability of the scheme if the user does not have this particular phone at transaction time. Moreover, this scheme only deals with transaction authentication, and does not focus on providing enhanced user authentication.

Full transaction authentication gives a bank the ability to ask the user to confirm her banking transaction to detect if MitB attacks are taking place and modifying the transaction *on-the-fly*. It is an essential step to enhance the security of online banking, as pointed out by Adham et al. [**?**]. The scheme we present in this paper achieves such goals without the need for additional hardware, as in CAP [**?**] or hPIN/hTAN [**?**], or for a long term secret stored in the user's smartphone. It also has the other features mentioned earlier, of covertly conveying information to the bank and supporting deceiving the adversary (honeyaccounts).

2.2 Use of Smartphones

Clarke et al. were the first to suggest the use of a camera-based device when connecting from untrusted computers [**?**]. While they did not explicitly discuss the use of QR codes, their paper is considered seminal in this approach of enhancing authentication. A number of follow-on proposals presented other camera-based schemes, using smartphones and other devices to improve authentication (see, e.g., [**?**,**?**,**?**,**?**,**?**,**?**], to mention a few).

Each one of these schemes suffers from one or more of the following shortcomings; (i) requiring an extra piece of hardware; (ii) storage of long-term secret on the smartphone; (iii) requiring a new registration process for associating the user's bank account with a particular smartphone; (iv) requiring the smartphone to have (network or cellular) connectivity to carry out the authentication process. The scheme we present in this paper does not suffer from any of these shortcomings.

2.3 Use of Deception and Covert Messages

The use of deception has shown a number of promising results in aiding computer defenses. Almeshekah et al. discussed the advantages of incorporating deception

into computer security defenses [**?**]. We incorporate deceptive elements in our scheme in two ways: (i) active MitM will be deceived such that he is forwarding the covert messages back-and-forth that sends an alarm to the service provider, (ii) we introduce honeyaccounts in our scheme to dismantle an attack before it takes place, and to gather information about the attacker's goals, objectives, and resources.

The *covert channel* term was introduced by Lampson in 1973 and defined as *"channels not intended for information transfer at all"* [**?**]. Such a channel has been extensively studied as a security vulnerability that undermines the security of a system and leaks out private information. The covert channel we are introducing in this scheme is observed to "not carry information" by the adversary and is created by design to *enhance* the overall security of the system. In this work we are overloading the term, although we see the functionality as similar.

Our method introduces the use of covert deceptive messages between the user and/or her client and the service provider. One of the choices of covert message is that the user is logging in as a response to an email; we discuss how this can be achieved in the next section. If the bank has no record of a recent communication, that response may trigger an enhanced defense, such as enabling read-only access. This would directly address many forms of phishing. Other messages can be automatically embedded by the user's client, such as the use of a public network.

Honeyaccounts are fake bank accounts that banks can use to lure attackers and deceive them into believing that they have successfully broken into the user's account at the bank. They provide an effective mechanism to monitor attackers' activities – to learn who is targeting a certain bank, and learn the other accounts being used to launder users' stolen funds. This information is usually gathered by banks during the forensic investigations following a money-theft episode (when it is too late to follow the money trail leading overseas). A user who covertly conveys to the bank her belief in the present transaction offers some hope of dismantling the financial infrastructure of a large-scale phishing campaign before it does real damage. We all experience situations where we *know* that an email is a phishing attempt, yet many of us limit our reaction to not falling prey to it — it would be nice to have an easy-to-use mechanism for conveying our belief and thereby triggering the deception mechanisms of the bank. The covert communication we propose can achieve this.

3 Technical Specification

This section discusses the technical specifications of our scheme. We show how to perform the initial setup at the server and seamlessly enroll users. We also discuss how the covert channel can be deployed within the authentication scheme. At the end of this section, we discuss some the potential enhancements that our scheme brings which can be incorporated in future work.

3.1 Attack Scenarios

There are many attacks against password-based authentication systems including the following common attacks.

- **Stolen Passwords.** The security of password-based authentication systems fundamentally relies on the fact that each user's password is only known to the user alone. When an adversary obtains the user's password he has the ability to *continuously* impersonate the user to the server, without any of the two parties noticing. Many attacks, such as phishing, keylogging, and shoulder-surfing are centered on the goal of obtaining users' passwords to gain unbounded access to their accounts.
- **Stolen Password Hashes File.** An adversary who obtains the passwords hashes file of many users can apply an offline cracking process (such as dictionary attacks) to retrieve the users' passwords from their hashes.
- **Poor/Easily Guessable Passwords.** When the user chooses an easily guessable password, an adversary can easily guess it and impersonate the user to the server.
- **Repeated Password Use.** A person may use the same passwords across multiple systems where a compromise against one system undermines the security of all other systems.

The focus of our design is primarily to address the first attack scenario. In addition, it provides a minor improvement to address the problem of cracking passwords.

3.2 Scheme – Setup

As depicted in figure **??**, there is no new registration required for bank customers, and the bank can roll out the deployment of the scheme either all at once, or progressively by selecting a specific subset of their customers (in which case a user who prefers the old system can easily be accommodated). In addition to a cryptographic one-way hash function H and a cryptographic message authentication code such as $\mathbb{HMAC}$, we use a one-way accumulator function A whose output is to have the same number of bits as H (so that the format of the bank server's password file does not need to be modified – only the nature of the bits stored changes).

As discussed by Fazio and Nicolosi, an accumulator function can be constructed such that it behaves as a one-way function [**?**]. In addition to the usual one-way property required of cryptographic hashes, a one-way accumulation of n items has the properties that (i) the order of the accumulation does not matter (i.e., any two permutations of the same n items give rise to the same result) [i.e. $A(x_1, x_2) = A(x_2, x_1)$]; and (ii) given a new item/s and the accumulation of a previous item $A(x_1)$, a new accumulation that includes the new item/s (as well as the old one) can be efficiently obtained without needing to know the previous item (x_1) which equals $A(x_1, new\_items)$. To illustrate the second property using an example, if we have the modular exponentiation of x_1 (g^{x_1}) and we want

to compute the new accumulation including a new item x_2, we compute this as $g^{x_1^{x_2}} = g^{x_1 * x_2}$. A real world realization of such a function can be done by using a modular exponentiation where the accumulation of x_1 can be implemented as $A(x_1) = g^{x_1}$.

As the most common ways of implementing such an accumulator A function involve modular exponentiation, it is typically the case that $A(x, y) = A(x * y)$ (where arithmetic is modular). In that case the security of A hinges on the Computational Diffie-Hellman assumption: That given $A(x_1)$ and $A(x_2)$ it is computationally intractable to compute $A(x_1, x_2)$ without knowing either x_1 or x_2. We give our presentation assuming the existence of such an A, without going into any details of how it is actually implemented (our scheme depends only on A's one-way property, its above-mentioned order-independence, and its above-mentioned incremental accumulation).

Recall that a user's entry in a traditional password file contains $h = H(passwd \mid\mid salt)$ and $salt$, where the purpose of the salt bits is to make a wholesale dictionary attack against all users harder (but it does not make it harder to attack an individual user, because the salt is in-the-clear). To switch to the new system, the bank simply replaces h with $A(h)$. This can handle users who select to remain in the traditional username/password authentication (in the obvious way). But replacing h by $A(h)$ is essential for users who select to switch to our proposed smartphone-based scheme, which we describe next.

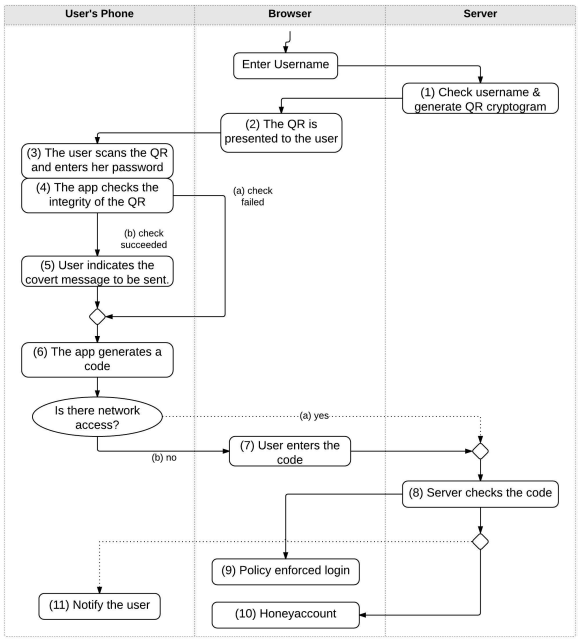

Fig. 2. Protocol Run

3.3 Scheme – Login

As usual, the login starts with the user entering her username on the computer. We assume that the smartphone has the needed app (which knows nothing about the user or the bank).

- The bank verifies that the username exists and, if so, generates a nonce R. Then it computes and sends the following information to the user's browser, encoded in a QR-code (recall that a QR code is an optically machine-readable two-dimensional barcode).
  - $A(R)$.
  - $\mathbb{HMAC}_{key}(A(R))$ where $key = A(A(h), R) = A(h, R)$.
  - The user's salt.
- The user scans the QR code using the smartphone app and inputs his password to the smartphone. The app computes $h' = H(password \mid\mid salt)$ and then generates the $\mathbb{HMAC}$ key by computing $A(A(R), h') = A(R, h')$ — the user's phone does not need a copy of R to make this computation. The $\mathbb{HMAC}$ is recomputed locally and then the app verifies that the received $\mathbb{HMAC}$ matches the $\mathbb{HMAC}$ it just computed. If the check succeeds (meaning the user entered the correct password and $h == h'$) the user moves into the next step of the protocol – phase 5. If the check fails there are two scenarios for what comes next: A *safe case* (branch a), and a *decoy case* (branch b). With the *safe case* the scheme continues to phase 5; in the *decoy case* the scheme jumps to phase 6 to send the MitM/MitB to a honeyaccount. In the latter case, the app can simply skip the covert messaging part if it detects a MitM/MitB impersonating the bank, and either terminate or continue with a honeyaccount. In this case, the failure of the $\mathbb{HMAC}$ verification can be treated as a special kind of covert message.
- In phase (5), the user is provided with the optional facility to covertly signal a simple message to the server. This covert messaging mechanism enables different behaviors from the current practice of "all-or-nothing" authentication and access. We give users the ability to choose from a fixed set of possible messages they could convey to the server. Giving users the ability to convey their level of trust in the computing or network facilities being used, e.g., using a public or a friend's computer, wireless network at an airport, etc. Later in this section, we show how these messages can be easily embedded in the code generated, in phase (6) of the scheme. Users can use this same facility to covertly request a limited-access login (e.g., read-only), in cases where they are following an email-solicited invitation to login to view an "important message." This covert message can alternatively be realized by other means than the above, such as those proposed by Almeshekah et al. [**?**].
- In phase (6), a one-time code is generated by the smartphone by computing the following accumulation:

$$y = A(A(R), h, msg_1, .., msg_i) = A(R, h, msg_1, .., msg_i)$$

The covert messages are conveyed by setting up the bit of any *covert message* (of the i possible messages) to one.

- In phase (7), the user types the generated code into the computer (copied from the smartphone screen). To make the code readable we can use base64 encoding and selecting the first n characters (the size of n is discussed later). Branch (a) of the previous phase, i.e. the existence of networking facility in the phone, will be discussed shortly.
- When the bank receives the code, in phase (8), it will check the validity of the code and whether a covert message has been signaled or not. It first accumulates into $A(h)$ the item R, if it matches the y sent by the user sent then the login succeeds (and the user did not convey a message), if it does not match y then the bank further accumulates (in turn) every possible covert message until the result matches y (or, if none matches, the login fails). In the *safe case*, if the bank receives a valid code with no message, phase (9) of the protocol is reached. However, if a message is sent, there are two possible options depending on the message:
  1. Take policy-specified action as per to the message conveyed before reaching phase (9). This can incorporate a variety of policies including the requirement of carrying out additional authentication measures or offer limited access. This gives service providers the ability to implement risk-based authentication and access control, and enforce a rich set of policies.
  2. Redirect the authentication session to a honeyaccount and, optionally, notifying the user of this access decision.

Length of code (y). As we will discuss below, the accumulator function is a one-way function and its output can be viewed as a random sequence of bits. As a result, the adversary succeeds if he can guess all the characters in this code. If we have 64 possible characters (including alphanumeric characters and symbols), the probability of guessing a single character is 2^{-6}. If we set the length of y to 5, the probability of guessing the code y is roughly equal to 2^{-30}.

In addition, as presented above, the calculation of y includes a random number R. As a result, the adversary gains no advantage by learning any previous runs of the protocol and the value of y as it is a one-way function of a number of variables including a random variable.

3.4 Incorporating Deception and Covert Communication

The introduction of covert channels in our scheme gives the user and app the ability to convey a number of pre-determined messages without the knowledge of any party positioning itself at any place in the communication channels. This can be done by appending a number of bits to the input of the accumulation function in step (6). To give an example, assume the protocol is designed to signal two different messages to the server; (i) msg_1 the user is accessing from a new wireless network, (ii) msg_2 the user selected read only access. When the app computes y in step (6) it can append two bits to the hash output as the

following; $y = A(A(R), h(passwd||salt)||msg_1||msg_2)$ where msg_1 and msg_2 are single bits that are set to 1 if the user want to signal this message and 0 if the message is not being signaled.

The multitude of applications that can utilize such a mechanism is large and it incorporates status communication as part of the authentication protocol. For example, the bank can take extra precautions if the user is authenticating from a new networking environment. Another example, is the user can signal duress if he has been threatened and forced to transfer money to other accounts. Duress can be signaled covertly, for example, by measuring rapid changes in the phone's built-in accelerometer where the user can subtly shake his phone during login. Another example to signal duress is when the user presses the physical volume buttons during the authentication process.

3.5 Security Analysis

Within our scheme when the bank sends $A(R)$, the only party that can successfully respond with y is one who knows the password and gets the smartphone to compute $h = H(password||salt)$ and thus the code y that is conveyed back to the server. This is true because an adversary who gets $A(h)$ and $A(R)$ is unable to compute $y = A(h, R)$ without knowing either R or h, neither of which is available to the attacker. Also note that, if the credentials database at the bank is leaked, no one can impersonate the user without cracking the passwords, as in traditional password schemes. One minor advantage this scheme provides is that cracking is slower for the adversary because of the introduction of the accumulation function A – it is significantly slower to accumulate every password in the cracking dictionary than to simply hash it.

Central to the security of our scheme is the fact that the only information of use to an adversary (the password) is entered on the cell phone and not on the client computer being used to remotely access the bank. The cell phone has no permanent record of any sensitive information. In addition, the bank's server never contains (even ephemerally) the user's password in the clear, providing a measure of defense against a snooping insider at the bank.

Finally, we point out that there are a number of additional security advantages of entering the user's password in a smartphone application instead of the browser:

- **The use of Software Guards.** Traditional password based-schemes ask the user to enter her password in the browser running on the client operating system. Current browsers are not self-protected, as identified in [**?**], and they are a complex piece of software that is exposed to many vulnerabilities. For that, our scheme uses a specific phone application that can have an intrinsic software protection against tampering as illustrated in [**?**,**?**].
- **Automated Trust Decision.** Adversaries using social-engineering attacks to lure users to give up their credentials, such as in the case of phishing, exploit the users' decision-making process by presenting them with legitimate-looking web pages. Our scheme aids users in making trust decision about

the authenticity of a web page mandating that the website provides a cryptographic proof of their knowledge of a shared secret; namely the password. This process is done in total transparency to the user and the user is only asked to capture the picture of a QR code.
This cryptographic proof can be computed by the web server without the need of explicitly storing the password value and, more importantly, without storing any information on the user's phone. This significantly increases the difficulty of social engineering attacks, such as phishing, as it reverses the game – demanding that the web site provides proof of authenticity before the user logs in.

– **Smaller Chance for Shoulder-Surfing.** Traditionally, users enter their passwords using a large keyboard where shoulder surfing is an easy task for adversaries. Asking the user to input their password on their phone increases the difficulty of such activity.

It worth noting that if the user logs-in to the service provider using a phone browser, our scheme cannot be directly used to scan the QR code as we discussed above. However, the basic protocol and feature can still be applicable with only a change in how the QR is input. This can be achieved by developing a browser extension that can automatically detect a QR code in the webpage and button on the corner of such codes to be clicked by users to launch the authentication app where the QR is automatically read. Nevertheless, the advantages of separating the service login, previously done on the computer, and the authentication process on the phone are slightly degraded. If the phone browser is infected with a MitB trojan, it would be easier to circumvent the security on the scheme as it can communicate directly with the authentication app. However, we note that most security sensitive transactions on a phone are done using dedicated apps that are hardened for a specific application. In addition, the underlying principle of using a covert channel presented in this paper can be incorporated in these dedicated apps.

3.6 Scheme – Enhancements

Full-Transaction Authentication – After the user logs in, the same steps can be repeated for every sensitive transaction with two main differences: (i) instead of sending the username, it is the transaction information that is sent, so that the QR code will contain additional information about the transaction details along with the $\mathbb{HMAC}$ and the user can verify those details on the app itself and make sure it is what they really want; and (ii) the covert message part can be eliminated, only keeping the part related to the failure of MAC checks. This part can be used, as we discussed before, to lure attackers who are launching MitB attacks manipulating transactions "on-the-fly."

Phone Connectivity – If the smartphone happens to have (optional) network connectivity (step (a) in figure **??**), it can spare the user the trouble of manually entering the code displayed on its screen, and send it itself to the bank's server (user sessions can be uniquely identified by the server using the nonce R).

Storage of Insensitive Information – The security of our scheme does not require the long term storage of any information in the phone itself. Nevertheless, we can benefit from storing information that can increase the utility of the covert communication. As an example, the app can store the name(s) of user's home network(s) and automatically send a covert message when the user is using a non-trusted network to login. Such knowledge gives service providers the ability to deploy a risk-based authentication. For example, when the user is using an untrusted network to login, a limited control can be provided and extra level of authentication can be enforced when powerful transactions are required.

4 Comparison with Other Schemes

In table **??** we evaluate the different schemes using the following criteria.

Requirement of phone enrollment — schemes such as CrontoSign and QRP [**?**] require the user to register her phone with the bank, i.e. phone enrollment. Such schemes store phone information, such as IMEI number, and use it as part of their protocol to achieve assurances about the user's identity. One of the major issue of tying the user's identity to his phone is that the user may lose his phone, forget it or run out of battery. In these circumstances, the user wants to be able to use an alternative phone to login to his account. If the user loses his phone he is vulnerable to the threat of impersonation until he reports the incident to every bank he banks with. In the case where he does not have his phone the usability of such a scheme becomes an issue as the user cannot login to his account anymore. This could result in lost business if the user moves to other banks that are using more usable schemes.

Our approach addresses these concerns in two ways. First, we allow users to use many phones without degrading the security of the scheme or asking the user to register all his phones. Second, we challenge the all-or-nothing assumption allowing users to fall back to other authentication mechanisms dynamically, possibly setting the privileges to only allow non-sensitive transactions.

Requirement of long-term secrets — many of the previously proposed schemes require the storage of long-term secret(s) either on the users' phones or on another piece of specialized hardware [**?**,**?**,**?**]. To the best of our knowledge, our scheme is the first scheme that provides full transaction authentication and user authentication that resist MitB without the need to store long-term secrets or require additional hardware.

Resisting MitB — a recent paradigm in banking Trojans is to bypass two factor authentication by launching MitB attacks that change transaction information on-the-fly. We compare the schemes in table **??** based on their resistance to MitB. When our scheme is used to authenticate transactions, as discussed in section **??**, a MitB attack can be defeated. This is because the MitB needs to send the modified transaction information to the bank, where an $\mathbb{HMAC}$ is

created. However, when the user verifies this information on his phone after scanning the QR-code he can see that the transaction details have been changed. He can click on a button to say that the details have been changed and a *deceptive* code can be generated. The MitB attacker would end up in phase (10) where they will be deceived.

	no phone enrollment	no long-term secret	resists MitB	no special hardware	no phone connectivity	compatible with existing
Our Scheme	✓	✓	✓	✓	✓	✓
CrontoSign [**?**]	–	–	✓	✓	✓	–
QR-Tan [**?**]	–	–	✓	✓	✓	–
hPin/hTan [**?**]	N/A	–	✓	–	N/A	–
QRP [**?**]	–	–	✓	✓	✓	–

Table 1. Schemes Comparison

Use of special Hardware — many proposals introduce a new piece of hardware to the authentication scheme to achieve a higher level of assurance and to verify banking transactions, such as the CAP scheme [**?**]. There are two major disadvantages with those approaches: cost and usability. As an illustrative example, Barclay's bank in the UK equipped users with special full-transaction authentication hardware, but ended up having to reduce the functionality to only partial transaction authentication because of many customer complaints. This degradation led to a number of security vulnerabilities [**?**].

Requiring phone connectivity — a number schemes are intended to maximize their usability by making the smartphone or the special hardware act on the users' behalf. In all the mechanisms we examined this comes with the cost of either requiring the phone to have network connectivity, which is not always possible, or requiring a direct communication between the users' computers and their smartphones, which hinders usability. In our proposal we share the same goals and enhance the usability of our scheme by giving users the ability to login even though they do not have any connectivity in their phone and without having to connect their phones to their computers.

Compatible with existing infrastructure — banks perceive security as an economic and risk reduction activity. Protocols that require radical changes to current infrastructure usually do not get adopted because of the associated high cost. In addition, the ability to dynamically fall back to traditional authentication methods is a preferred property giving banks the ability to dynamically deploy their new scheme and progressively enroll their users. This is why we use this as a comparison factor with other schemes.

5 Conclusion

We propose an authentication mechanism that has many attractive features, including compatibility with deployed authentication infrastructure; flexible use of smartphones without requiring phone registration or storage of permanent information in the phone; without any requirement of phone connectivity (i.e., using the phone as a computational device rather than as a storage or communication device); resistance to many common forms of attack; and a facility for user-friendly (pull-down menu on the cell phone app) covert communication from the user to the bank. The covert communication in turn makes possible different levels of access (instead of the traditional all-or-nothing), and the use of deception (honeyaccounts) that makes it possible to dismantle a large-scale attack infrastructure before it succeeds (rather than after the painful and slow forensics that follow a successful phishing attack).

References

1. M. Adham, A. Azodi, Y. Desmedt, and I. Karaolis. How to Attack Two-Factor Authentication Internet Banking. In *Financial Cryptography*, 2013.
2. M. H. Almeshekah, M. J. Atallah, and E. H. Spafford. Back Channels Can Be Useful! - Layering Authentication Channels to Provide Covert Communication. In *Security Protocols XXI*, Lecture Notes in Computer Science, 2013.
3. M. H. Almeshekah and E. H. Spafford. Planning and Integrating Deception into Computer Security Defenses. In *New Security Paradigms Workshop (NSPW'14)*, Victoria, BC, Canada, 2014.
4. American Banking Association (ABA). Popularity of Online Banking Explodes, Sept. 2011.
5. H. Chang and M. J. Atallah. Protecting software code by guards. In *Security and privacy in digital rights management*, pages 160–175. Springer, 2002.
6. D. Clarke, B. Gassend, T. Kotwal, M. Burnside, M. Van Dijk, S. Devadas, and R. Rivest. The untrusted computer problem and camera-based authentication. In *Pervasive Computing*, pages 114–124. Springer, 2002.
7. S. Drimer, S. J. Murdoch, and R. Anderson. Optimised to fail: Card readers for online banking. In *Financial Cryptography and Data Security*, pages 184–200. Springer, 2009.
8. I. Drokov, E. Punskaya, and E. Tahar. System and Method For Dynamic Multi-factor Authentication, 2006.
9. P. Falcarin, C. Collberg, M. Atallah, and M. Jakubowski. Software Protection. *Software, IEEE*, 28(2):24–27, 2011.
10. N. Fazio and A. Nicolosi. Cryptographic accumulators: Definitions, constructions and applications.
11. N. Harini and T. R. Padmanabhan. 2CAuth: A New Two Factor Authentication Scheme Using QR-Code. *International Journal of Engineering and Technology*, 2013.
12. B. W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.

13. Y. Lee, J. Kim, W. Jeon, and D. Won. Design of a Simple User Authentication Scheme Using QR-Code for Mobile Device. In *Information Technology Convergence, Secure and Trust Computing, and Data Management*, pages 241–247. Springer, 2012.
14. Y. S. Lee, N. H. Kim, H. Lim, H. Jo, and H. J. Lee. Online banking authentication system using mobile-OTP with QR-code. In *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*, pages 644–648. IEEE, 2010.
15. S. Li, A. R. Sadeghi, S. Heisrath, R. Schmitz, and J. J. Ahmad. hPIN/hTAN: A lightweight and low-cost e-banking solution against untrusted computers. In *Financial Cryptography and Data Security*, pages 235–249. Springer, 2012.
16. K.-C. Liao and W.-H. Lee. A novel user authentication scheme based on QR-code. *Journal of Networks*, 5(8):937–941, 2010.
17. M. Mimoso. Two-Factor Authentication No Cure-All for Twitter Security Woes.
18. D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen. RFC 4226 - HOTP: An HMAC-Based One-Time Password Algorithm. Technical report, IETF, 2005.
19. D. M'Raihi, S. Machani, M. Pei, and J. Rydell. RFC 6238 - TOTP: Time-Based One-Time Password Algorithm. Technical report, IETF, May 2011.
20. S. Mukhopadhyay and D. Argles. An Anti-Phishing mechanism for single sign-on based on QR-code. In *Information Society (i-Society), 2011 International Conference on*, pages 505–508. IEEE, 2011.
21. D. Pintor Maestre. QRP: An improved secure authentication method using QR codes. 2012.
22. Risk Analytics. $70 Million Stolen From U.S. Banks With Zeus Trojan.
23. G. Starnberger, L. Froihofer, and K. M. Goeschka. QR-TAN: Secure mobile transaction authentication. In *Availability, Reliability and Security, 2009. ARES'09. International Conference on*, pages 578–583. IEEE, 2009.