Delayed password disclosure

Markus Jakobsson*
Palo Alto Research Center,
3333 Coyote Hill Road,
Palo Alto, CA 94303, USA
E-mail: markus@markus-jakobsson.com
*Corresponding author

Steven Myers

School of Informatics,
Indiana University,
Bloomington, IN, USA
E-mail: markus@indiana.edu
E-mail: samyers@indiana.edu

Abstract: We present a new authentication protocol called Delayed Password Disclosure (DPD). Based on the traditional username and password paradigm, the protocol's goal is aimed at reducing the effectiveness of phishing/spoofing attacks that are becoming increasingly problematic for Internet users. This is done by providing the user with dynamic feedback while password entry occurs. While this is a process that would normally be frowned upon by the cryptographic community, we argue that it may result in more effective security than that offered by currently proposed 'cryptographically acceptable' alternatives. While the protocol cannot prevent *partial* disclosure of one's password to the phisher, it does provide a user with the tools necessary to recognise an ongoing phishing attack, and prevent the disclosure of his/her entire password, providing graceful security degradation.

Keywords: decisional and static Diffie-Hellman; doppelganger; oblivious transfer; OT; password authenticated key exchange; PAKE; phishing; secure user interfaces.

Reference to this paper should be made as follows: Jakobsson, M. and Myers, S. (2008) 'Delayed password disclosure', *Int. J. Applied Cryptography*, Vol. 1, No. 1, pp.47–59.

Biographical notes: Markus Jakobsson is Principal Scientist at Palo Alto Research Center, and Adjunct Associate Professor at Indiana University. He is a Visiting Research Fellow of the Anti-Phishing Working Group, a Founding Member of the RSA eFraud Forum, Founder of the security startup RavenWhite, Advisor to the Financial Services Technology Consortium and a Consultant to the financial sector and of Crimeware: Understanding New Attacks and Defenses (Symantec Press, 2008). He has served as Principal Research Scientist at RSA Laboratories, Adjunct Associate Professor at New York University and a Member of the technical staff at Bell Labs. He is an Editor of *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft* (Wiley Press, 2007). He received his PhD from the University of California at San Diego in 1997.

Steven Myers is an Assistant Professor in the School of Informatics at Indiana University, where he is also a Member of the Center for Applied Cybersecurity. His research interests are in all areas of cryptography, computer and systems security with a specific interest in phishing. He has written several papers, lead panels, and given invited talks in fields ranging from Cryptography and Computer Security to Distributed Systems and Probabilistic Combinatorics. Recently he co-edited the book *'Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft'*. He received his PhD from the University of Toronto in 2005.

1 Introduction

Apart from the direct damage phishing is doing to the financial industry, it is also seriously threatening to halt the expansion of e-commerce, due to the erosion of trust among many computer users. While legislative approaches may keep some would-be phishers out of the game, it is unlikely that law enforcement alone can cause a noticeable dent in the phishing statistics, making technical countermeasures important. However, given that phishing is *both* a technical and a social problem, such countermeasures are far from straightforward. Moreover, given the complexity of the

problem, it is not likely that there would be *one* solution that would address it in its entirety. Instead, it is likely that many different solutions will be needed to lessen the success rate of various types of attacks.

An important first step is therefore to understand the taxonomy of phishing attacks, allowing us to tailor countermeasures for each class of attacks. Not counting malware based attacks (such as spyware or keyboard loggers), one can notice that most phishing attacks consist of a *delivery* component (often using spam techniques) and a *mimicry* component. The latter presents the victim with information intended to cause him or her to divulge particular user credentials, while being under the impression that he or she accesses the resource that is being impersonated.

One of the main difficulties of preventing mimicry is what one may term the *security gap* between users and their machines. More specifically, all user decisions are made given the information users are presented by their computers. Practitioners and researchers alike agree that it is a difficult problem to determine how to convey to a user information relating to his or her security status – in particular in the light of the possibility of the adversary attempting to present the similar but invalid indications to targeted victims. For example, almost all phishers use the logos of the organisations they are trying to impersonate, and some cause a small lock (identical to the SSL icon) to be presented in the body of web pages that are *not* secured by SSL. Users often do not realise that the lock icon is in the wrong location, and have no way of determining that the logos were used by somebody other than their rightful owners. In fact, many legitimate companies' sites add to this confusion, as they place similar lock logos on their web pages in numerous spots, in an effort to convey to customers the belief that their site is secure. Another reason for the security gap is that users do not notice the presence of warnings or the absence of reinforcing security information, as supported by Jakobsson and Ratkiewicz (2006) and Schechter et al. (2007).

We believe that mimicry will become increasingly sophisticated, with the functionality of entire sites being emulated by attackers, and in particular, with any type of 'secure login' windows being mimicked, 'even if said windows are provided by the browser'. This will circumvent the security of techniques such as Password Authenticated Key Exchange (Bellare et al., 2000; Katz et al., 2001; MacKenzie et al., 2000), which (quite naturally) base their security guarantees on *being used*. We call this new type of attack a *doppelganger window attack* as it evades security requirements by looking the same as (potentially) secure interfaces but without offering the 'behind-the-scenes' functionality of these. In other words, the attacker strives to become a *doppelganger* – or identical twin – of the various interfaces he wants his victim to see.

In this paper we introduce the notion of doppelganger attacks, and offer a first treatment of how to defend against these. This is done in the context of *promiscuous users* – that is, users who may operate from a large number of semi-trusted[1] terminals, some of which may not have been used before. We note that if one instead assumes non-promiscuity, the problem is simplified, as the user's *computer* can automatically verify (by means of stored state certificates, digital signatures, etc.) that a previously visited site is not impersonated. Indeed, under this assumption, no user login is needed: the machines can verify each other's identities by means of cryptographic authentication methods before access is granted (although in such scenarios a password may still be desirable to ensure the legitimate user is the ones currently using the machine). Thus, such a setting shifts the emphasis to that of protecting against malware (as is needed to some extent for all approaches) and securing the initial access to a user machine.

One approach to defend against such doppelganger attacks is that of a secure communications pathway. Traditionally, these have been used for the initial access (i.e. logon) of the user's machine at the operating system level, but may in principle also be used for any user authentication process, even at the application level–although, this would require OS support. This corresponds to a solution in which a trusted third party is placed on a terminal (normally a portion of the operating system), and is used to guarantee that all login attempts are made using the *intended protocol*, and not some fake masquerading interface. It also ensures that no other process gets access to the credentials. For example, imagine a solution in which a user must always – when presented with a secure login window – press some combination of keys that moves the computation into a 'safe state' in which only very restricted authentication functionality and its interface are made available. This would shift the problem to that of securing the operating system, and allow the secure use of standard mutual authentication techniques, such as the previously mentioned PAKE protocols. However, in the absence of such a solution, alternative approaches (such as ours) are necessary.

1.1 Our contributions

We propose a protocol that permits a user interface that provides users with visual character-by-character feedback as they enter their passwords, allowing users to stop entering their password if they obtain feedback that they do not recognise – a sure sign of interacting with the wrong site. While there are numerous ways in which the interface to such a protocol could be implemented, our goal is to provide a secure protocol on which such interfaces can be built. As an example, with our protocol, one could require that passwords are entered by pointing to keys of an online keyboard over which the feedback images are displayed, or to confirm with the mouse that each image displayed is correct after it was entered in the keyboard; but these are just two interfaces that could be hung off the same protocol. Finally, we point out that while users will be required to recall passwords, as they traditionally have, they need only to recognise feedback, a cognitively much simple cognitive task than recall.

Our approach, which is based on Oblivious Transfer (OT) and PAKE, is proven secure in several critical ways based on standard cryptographic assumptions and models. A contribution of potential independent importance is a blinding technique used to reduce the costs of communication and computation of the OT component of the solution by introduction of one extra move in which the client sends the server a blinded request, the response to which is later unblinded to obtain the actual response. This technique allows a high degree of security (quantified by the number

of possible images that can be selected for each password character that is entered) at the same time as a reasonably efficient implementation.

2 Related work

The rapid rise of phishing attacks and their potential to have large negative effects on e-commerce has resulted in a significant number of researchers trying to solve the phishing problem. The approaches have varied widely, which has appropriately given the fact that phishing is at heart a social engineering attack, and thus can take on many different guises. We briefly review some of the main works in this area.

Chou et al. (2004) use a system that evaluates a given web page and comes up with a 'phishiness' index to indicate the likelihood that the web page in question is that of a phisher. The index is computed based on factors including, but not limited to: similarity of URL to known phishers' sites, the inclusion of official logos from official sites and requests for passwords and credit cards.

Several commercial efforts, among them those by Microsoft and eBay, involve browser extensions to flag blacklisted sites (where an updated blacklist is frequently made available for automated download). A related but academic effort is the Trust-Bar construction by Herzberg and Gbara (2004), which associates logos with the public keys of the certificates of visited sites. The hope is that displaying the logos will create a better conceptual connection between the organisations certifying that a site can be trusted and the trusted site. Further, the use of logos allows for the development of corporate branding for certificate authorities, creating a valuable asset for which they will have incentive to protect, by not improperly issuing certificates in improper situations.

When a user can knowingly trust the user interface he or she is using, then traditional cryptographic PAKE protocols can be used to ensure security, as the phisher cannot simply bypass them. Therefore, a key issue in overcoming doppelganger attacks is to produce a trusted path from the user to the server, when possible. An excellent review of trusted paths is available in Emigh (2005). A way to circumvent the traditional problem of trusted path was recently suggested by Kuo et al. (2004) their approach involved the use of an auxiliary device (a cell phone) to maintain trusted state. Dhamija and Tygar (2005) also approach the phishing problem by trying to address phishing by creating a trusted path between the user interface and the user. In their proposal, to establish trust in the user interface, a user must select a specific image which will be superimposed onto all dialogue boxes presented by the browser. Since the selected image is known only to the user (and possibly provided by the user), the phisher cannot duplicate such a dialogue box. For this solution to be used, the user must have a preestablished secret (the user's image) with the browser, and so this technique does not allow for promiscuous browser use.

Clearly, the notion of trusted path is very relevant to defend against phishing and doppelganger attacks. The trusted path problem is closely related to the *trusted computing* problem (see (Smith, 2005) for a comprehensive discussion). While trusted paths are typically concerned with how to secure data input (e.g. by standardised and secure interfaces) trusted computing instead is concerned with controlling what processes are running on a given machine, and what resources they have access to. Typical trusted computing efforts do not address the doppelganger problem, since an attacker may try to deceive a user to perform some action (such as inputting his password) using a web browser window. Trusted computing controls what applications are run, and must for practical reasons allow the execution of web browsers; in typical trusted computing scenarios the contents of web pages viewed by the browser are not verified or validated. Note that it is non-trivial to automate the interpretation of content, as can be seen by the failure of commercial efforts to block spam.

Passmark™ (also branded as *SiteKey*) is a product of RSA Security (EMC), and has recently been deployed by the Bank of America. This product attempts to fight phishing by building on the traditional username/password interface by having the site authenticate back to the user. It does so by first recognising client machines by means of previously saved cookies, and by requesting the user to enter his username. If the user and his machine are recognised then Passmark causes a user-specific image to be displayed, after which the user can enter his password. Users are trained not to enter their password unless they recognise the image displayed to them. The product also has a server-side component that analyses login attempts and usage patterns, looking for fraudulent activity.

While Passmark is a promising approach, the client-side portion has some drawbacks. Firstly, in the presence of cookies, Passmark protected sites display an image to the user, but this constitutes placing a secret on the user's computer; in such cases stronger cryptographic mechanisms surely could be used. Secondly, given that traditional cookies are not resilient to pharming, their techniques do not seem to provide any protection against such attacks. Finally, alternative identity verification methods are needed when users do not accept cookies or when users migrate between machines (i.e. the case of promiscuous user). In such cases, the user is not presented with the image, but is instead authenticated through out-of-band communication or by responses to contextual information specific to the user (i.e. the user's bank account balance or the user's mother's maiden name). This approach, therefore, is vulnerable both to cookie theft and to attacks aimed at inferring private information such as mother's maiden names from available information (Griffith and Jakobsson, 2005). The use of back-up questions in the absence of cookies also gives rise to attacks such as the one described by Soghoian (2007). Additionally, user studies by Schecter et al. (2007) have shown that the approach may not succeed in bridging the security gap between users and their machines: Subjects in a user study were found not to react to the absence of Passmark images. Further, the subjects in the study were aware of participating in a phishing study, which is known (see, e.g. (Anandpara et al., 2007)) to introduce a bias. However, a similar observation was made in the context of the 'eBay user greeting' in Jakobsson and Ratkiewicz (2006), where subjects were not aware of participating in a study, much less a phishing study.

A similar technique to Passmark is that of Yahoo! Site seal; this is a technique by which users can upload text or images to a server which then displays these when the user returns to the site. Like Passmark, this is a technique reliant on cookies and elements stored in the user cache (Yahoo! Inc., 2006), but instead of identifying the *user* (which requires asking for the user name), it only recognises the *browser* installed on a given machine. While the interface of Site Seal is similar to that of Passmark it is not identical, and it is unclear, although likely, that it would suffer from the same lack of user recognition that current Passmark interfaces suffered, as previously discussed.

While it remains possible that users of our proposed Delayed Password Disclosure (DPD) system would also fail to notice the absence of their correct feedback images, we note that such issues are critically dependent on the user interface that is used to interface with the protocol, and that an appropriately designed user interface can address this problem. We note that the design of such an interface is the purview of HCI experts, and therefore out of the scope of this paper. Our goal is to design a secure protocol upon which such an interface can be draped. Our emphasis is on the design of a cryptographic protocol that meets realistic efficiency requirements and which supports a user interface that helps defend against doppelganger attacks, without the need for local storage such as cookies.

3 Doppelganger attacks

While we have briefly mentioned doppelganger attacks, in this section we formalise the notion, and discuss some of the difficulties in defending against strong online doppelganger attacks.

In a doppelganger attack, an attacker controls one or more windows on a user's machine, and produces an output that duplicates the appearance (and apparent functionality) of a given target site. The attacker aims to make a victim believe that he or she is interacting with the given target site, while he or she instead is interacting with the attacker. The goal of this is to allow the attacker to capture the victim's credentials, and later enable the attacker to impersonate the user to the legitimate site. We will consider two classes of doppelganger attacks.

3.1 Off-line doppelganger attacks

We may assume that the attacker has one or more accounts with the target site. The attacker is permitted to communicate with the target site a polynomial number of sessions in order to learn the behaviour of the target site, and collect other information necessary to duplicate the appearance of the target site. Once this process is completed, the attacker constructs a doppelganger site, and tries to cause the user to enter her credentials (associated with the target site) as input to the doppelganger site. The attacker may later connect to the target site in order to attempt to impersonate the victim (using the harvested credentials). The attacker is assumed not to have open sessions with the target site and with users at the same time.

Let us now consider the need for interactivity to address *off-line* doppelganger attacks. Suppose to the contrary that there is no feedback to the user *during* password entry. Any feedback given to the user *before* password entry can be duplicated by a off-line doppelganger attack – recall that there are no shared secrets between users and their machines, nor trusted input pathways. This attack is mounted by having the attacker enter the username of the client into the authentic site's page, storing the resulting display feedback, and ceasing communication with the authentic site. Now if the same user accesses the doppelganger site, the attacker looks up the display information and presents a doppelganger display to the user. Since the display is identical (or similar) to the display that the user expects to see, she enters her password to the user. In contrast, if authenticating information is only displayed *after* the password is entered, then an attacker will have the user provide her username and password, and will simply not provide the appropriate feedback. The user at this point may realise that she was attacked and this may be beneficial, but her password has been disclosed to the attacker at this point.

3.2 Online doppelganger attacks

The attacker is permitted all of the benefits of an off-line attacker, but is not required to terminate communication with the authentic site once it starts trying to convince users about the authenticity of the doppelganger site. Thus, this is a type of Man-In-the-Middle (MIM) attack. We distinguish it from tradition cryptographic MIM attacks, because here the attacker is interacting at the interface level with the victim, as opposed to the protocol level that cryptographers are concerned with (i.e. the victim is not running security protocols that she might think she is, but rather interfacing with a maliciously controlled interface that tricks the victim into thinking she is running such protocols).

Given our assumption that the user has no shared secret with her machine, and that there is no piece of trusted real estate on her display, it follows that there is no image that her machine can display that cannot be duplicated by the attacker. Suppose a user visits the doppelganger's site, the doppelganger then quickly visits the authentic site, and based on its appearance draws the same picture on the user's display as was shown to the attacker on his. Note that all of the information sent by the user is available to the attacker as there are no security protocols enabled on the doppelganger site; similarly, all information sent by the authentic site becomes available to the attacker, as he is the apparent end-point of the communication as far as the authentic site is concerned.

In effect, in the above attack, we have the attacker controlling what is displayed on the user's monitor, and everything she inputs into the machine is sent directly to the attacker. Thus, we can think of the user as effectively interfacing with a malicious machine, and there is no hope of protecting the input of the user at this point, as we could simply imagine that the malicious machine has an input-logger that captures the user's confidential information. In fact, if the doppelganger code is run on the victim's

machine, then it becomes little more than a special-purpose keyboard logger. If we assume that the doppelganger code is run on another machine (that may be easier to compromise than the victim's machine) then it may be possible for the service provider to detect anomalies in terms of the geographic mapping to IP addresses, and the number of sessions started from the range of IP addresses belonging to the machines controlled by the attacker. This is not studied herein, though, as we focus on the off-line attack alone; while this is arguably easier to defend against, such an attack is also easier to mount.

4 A high-level view of our goals

If there were a *truly concurrent* way for the human user (represented by the client machine) and the server to verify each other's credentials without leaking them in case of failure, then the problem considered in this paper would no longer exist. While it may appear that the problem of concurrent signature exchange (e.g. (Blum, 1983; Franklin and Reiter, 1997; Garay, 1999)) would be closely related to this issue, there are notable functional differences relating to when either *user* (and specifically not their machine) would *learn or enter* information. In the absence of true concurrency, we propose a technique to approximate concurrency in this particular context. This is done by letting the server machine provide gradual feedback on the credentials entered in the client machine; this feedback is presented to the human user, allowing the same to stop the login process if invalid feedback is noted.

Before describing our approach in more detail, it is important to consider the potential danger presented by providing gradual feedback. Among cryptographers, it is a well-understood design principle that one should maximise the entropy of the distribution from which secret credentials are drawn, by not processing these little by little. In particular, if a server were to verify an entered password character by character as these are entered, and halt if an incorrect character is seen, then this will very clearly allow for an interactive divide-and-conquer approach to determining the password of a victim. We are well aware of this potential threat, and address the problem as follows.

4.1 Gradual and flexible feedback

We place two requirements on our solution:

1 The server should not obtain any partial credentials during the execution of the protocol, but only after the human user (of the client side) has approved all gradual feedback material. This is to prevent the scenario where an attacker, acting as a legitimate sever running the legitimate protocol, learns a few bits of a user password in spite of not knowing the correct visual feedback.

2 Each feedback item should be dependent on the most recent prefix of the credential, that is, on all the characters that have been entered. This is both to maximise the min-entropy of the feedback, which in turn increases security, and to secure against temporary failures to recognise invalid visual feedbacks. Namely, and as we will later describe in more detail, we assume that users may occasionally fail to recognise incorrect visual feedback, and we want to make sure that the first mismatch between the user-entered password and the version stored by the server causes all consecutive feedback to be incorrect.

These two requirements may appear to be in contradiction with each other, since the first suggests that the server cannot learn any information until the protocol completes, whereas the second states that mistakes to recognise invalid feedbacks are aggregated. However, this apparent contradiction can be resolved by the use of OT techniques; these allow the client to index (using the credential prefix) a *database* of feedback items that is unique to the user in question. While we could employ simpler techniques not involving OT, the use of OT allows us to defend against a more realistic threat, corresponding to a tiered adversarial model.

4.2 Side channel attacks

An attacker can only improve his chances of success (beyond that of guessing the password) by either performing shoulder surfing on a victim (which is excluded by our adversarial model, as will be detailed next), or by interacting as a man-in-the-middle with both the victim and the server a large number of times.

In the former case, the attacker would have to start a session with the server; guess a first credential prefix and compare the resulting feedback to the observed feedback (retrieved through shoulder surfing). This would have to be done repeatedly, which would provide a practical indication of attack. We note that typical implemented authentication systems allow for only between three and ten mistakes before special action is taken; in this context, this first attack is less serious than the doppelganger window attack would be in the absence of our countermeasure.

In the latter case, the attacker would attempt to infer the expected sequence of feedback items by populating an entire malicious database with the images retrieved through interactions with the legitimate server, and determine from the user's reaction whether this was the correct image or not. Assuming, for simplicity, that the password alphabet has only 26 characters (a clear lower bound), then the attacker could interact with the server 26 times to retrieve all possible feedbacks for the first character of a password for a user it wishes to attack. The attacker would then create a database in which only half of the retrieved images correspond to the correct images returned by the legitimate server; the attacker would then trick the user to attempt to log in to his 'half correct' service. If the user detects the attack (and halts input), then the attacker knows the password character corresponds to the modified half of the database, and otherwise the attacker learns the character corresponds to the correct half of the database. By performing this attack some $\log_2 26$ rounds with different correct subsets of images, the attacker is able to determine what the first password character is. Note that this would only require $26 + \log_2 26$ interactions with the server and client and the attack is guaranteed to learn the first character of the

password. This could then naturally generalised to retrieve the remaining characters of the password. All in all, this would result in an attack that uses only $m(26 + \log_2 26)$ calls, where m is the number of characters in the password. This is substantially better (for the attacker) than the best possible attacks on PAKE protocols. There, the same number of interactions with the server and client would only be able to successfully share a key with either party with a probability of at most $(26m + (\log_2 26)m)/(26^m)$. Here, the probability corresponds to the number of interactions over the size of the dictionary of passwords (Note that we are assuming a uniform distribution of passwords over the dictionary). However, while *theoretically speaking*, this is a truly devastating attack, it has very limited practical applicability as typical users would be expected to react to a threat of this type before any meaningful amount of data has been leaked, and most servers now halt interaction with given clients after a small number of interactions which fail to properly authenticate. Further, a doppelganger attack on current PAKE or SSL protocols would allow the attacker to retrieve the entire password with one faulty interaction, whereas our protocols negates the possibility of off-line doppelganger attacks without many interactions.

We provide a solution that allows for a careful analysis, while operating in a threat model that is practically oriented. We use OT techniques to allow the exchange of feedback and credentials, without leaking the credentials. If the human user accepts the feedback items, then a traditional password-based key exchange is performed on the then-fully-entered credential. While the traditional use of OT techniques would render our approach impractical to the extent that it would not be deployable, we are providing some new efficiency improvements that bring down the costs to a range where the technique offers substantial promise. However, for common deployment in settings where computational resources are scarce, further improvements may be necessary; we hope that our proposed design can spur follow-up work that lowers the costs to the extent that the associated techniques are *truly* practical.

5 The model

5.1 Participants

We consider the following participants: human users (also referred to plainly as *users*); user machines; target sites (service providers with established relationships to users) and the adversary. We assume that users have preestablished shared secrets with target sites (passwords to be remembered and feedback to be recognised), and that these are relatively short (i.e. not of sufficient length to allow secure usage as cryptographic keying material). There are no preestablished secrets between user machines and target sites.

5.2 Adversary

We consider a remote, networked adversary; while the problem of shoulder surfing is a realistic threat in some contexts, it is rather unlikely in the context of phishing attacks. In practice, we also believe that users are also more likely to understand and protect themselves from shoulder surfing attacks than phishing attacks; this is because shoulder surfing exists in the real world, where people have a strong intuition for such things, as opposed to phishing which relies on a completely artificial security context provided by the browser's user interface.

The adversary is assumed to have several cryptographic as well as doppelganger abilities. Cryptographically, we must allow the adversary to be a passive adversary that can listen in on valid authentication transcripts between users and servers. Next, as an active adversary it ought to be able to execute the DPD protocol as either a malicious server, or client, as failure to protect against such attacks could render the protocol useless. Finally, an even stronger model should be considered in which the malicious Man-In-The-Middle (MIM) adversary that can control all network traffic, and concurrently manipulate the traffic of many simultaneous authentication sessions. Due to the interactive nature of the protocol, traditional PAKE security models are sufficient, and we leave the development of such a model to future work. Without an appropriate model, we do not prove our protocol secure in this setting although we conjecture it should be, for some reasonably strong model. We discuss some of the issues of such a model and our protocols security against MIM adversaries in Section 8.3. Finally, we note that such a MIM adversary is substantially more powerful than those currently enjoying success with phishing attacks, We also note that while the online MIM model is more commonly considered in much of the literature, the off-line model has practical merit as various risk assessment tools run on the back-end of online banks have the potential of detecting plausible man-in-the-middle attacks given odd network traffic patterns often associated with MIM attacks; and while a well masqueraded man-in-the-middle attack can be run by first placing malware on the client machine (after which the intermediary would run on this machine), such an attack automatically bypasses all known client-side security measures by virtue of its strength.

From the perspective of a doppelganger attacker, the adversary is assumed to be able to direct selected victims from a target site to a site controlled by the adversary; this may be done either at the beginning of, or during,[2] a session. The adversary may also control selected processes running on the user machine (as described below), and may interact with the target site before or during an attack (as detailed in Section 3).

5.3 User machine

We do not assume that the user machine has any *stored state* relating to previous login sessions (for the given user or others), nor that the target site has any information relating to any potential public key associated with the combination of the user and the target machine.[3] Furthermore, we model the machine as a *semi-trusted* node:

1 we assume that all processes are isolated from each other, and cannot access each other's storage, input or output, except for any information transmitted over the network

2 competing processes may run independently of each other, may display information to the user, and will receive any user input entered in their respective windows.

Finally, we do not assume the existence of so-called secure chrome, or any other secure monitor real estate.

5.4 User assumptions

It is important to model users in a realistic manner, as their behavioural characteristics are an important part of the problem. Whereas traditional cryptographic protocol research does not consider the human factor, it is important to do that in our setting. Our initial problem statement revolves around the problem for typical users to identify the correct user interface, especially so in the context of adversarial mimicry. Users often fail to notice security information communicated to them (Wu et al., 2006). They are better at noticing the presence of incorrect information (Jakobsson et al., 2007) than the absence of correct information (Jakobsson and Ratkiewicz, 2006), but commonly make mistakes of either kind. We rely on users to somewhat reliably detect the presence of incorrect information and the absence of correct information; we do not quantify the probability with which this is done, as we do not have data supporting what is realistic to assume. Further, any such data would be severely dependent on the interface draped over the protocol, and, as we have stressed, the design of such an interface is a completely separate problem.

5.5 Computational assumptions

We assume that we are functioning in the random oracle model, which assumes that before the protocol begins, a random function $H : \{0,1\}^* \rightarrow \{0,1\}$ is made available to all of the parties. While its known that arbitrary protocols proven secure in the random oracle model cannot be securely implemented (Bellare et al., 2004; Canetti et al., 1998; Goldwasser and Kalai, 2003), in practice it has shown to be an effective heuristics, and the RSA-OAEP subprotocol of TLS is already reliant on the assumption, so from a practical point of view this assumption is the same as the one currently made. Other than this, the standard cryptographic assumptions apply: all parties are limited to computing in probabilistic, polynomial time. We note that the random oracle is needed to provide an efficient OT algorithm that satisfies stronger security requirements than are normally required. In essence, we require a limited form of non-malleability against a man-in-the-middle attacker.[4] Further, many PAKE protocols make use of the Random Oracle model (the exception is the work of Katz et al. (2001), but this protocol is not widely deployed). Finally, our protocol relies on the traditional Decisional Diffie-Hellman (DDH) assumption, and a new stronger but seemingly reasonable assumption based on it and related to that Static Diffie-Hellman problem, where the adversary has limited access to an exponentiating oracle. It addresses the issue protocols that effectively give adversaries access to static DH oracles $O^y(x) \stackrel{\triangle}{=} x^y$ that raise group elements to the power y, when only g^y is known by the adversary. Examples include El Gamal (1985) in the case of chosen ciphertext attacks and the Ford-Kaliski Key retrieval protocol (Ford and Kaliski, 2000). This has been studied to some extent in the computational Diffie-Hellman case by Brown and Gallant (2004), who show connections between it and the hardness of the underlying Discrete-Logarithm problem. However, the issue has been left seemingly unstudied in the stronger decisional case. In particular, we posit that having access to such an oracle O^y for a small constant number of queries, say t, is helpful only in computing t Diffie-Hellman values, and is not of any other help in distinguishing even a $t+1$st Diffie-Hellman value from random, even if the values $g^{x_1}, ..., g^{x_t}$ of interest to the adversary were known in advance. This is formalised below. We believe that this new assumption may of interest in its own right, and deserves further study.

Assumption: Let *IG* be an algorithm that takes as input a unary security parameter, and outputs a cycli.e. group G with generator g and it order $\ell = | < g > |$ (i.e. say by generating a prime $p = 2\ell + 1$ for prime ℓ that is n-bits long, and outputting an element of the cyclic subgroup of Z_p^* that is of order ℓ). The Limited Static Diffie-Hellman (LSDDH) assumption holds if for all probabilistic polynomial time algorithms A, for all constants $c > 0$, constants $s, t \in \mathbb{Z}^+$ $(t < s)$, and for all sufficiently large n:

$$\left| \Pr_{\substack{x_1,...,x_s,y \in_U Z_\ell \\ j \in_U Z_s}} \left[\begin{array}{c} A^{O^{y}{}_t}\left(G, g, g^{x_1}, ..., g^{x_s}, g^y\right) \rightarrow \sigma; \\ A\left(\sigma, g^{(x_j y)}\right) \end{array} \right] - \Pr_{\substack{x_1,...,x_s,y,r \in_U Z_\ell \\ j \in_U Z_s}} \left[\begin{array}{c} A^{O^{y}{}_t}\left(G, g, g^{x_1}, ..., g^{x_s}, g^y\right) \rightarrow \sigma; \\ A\left(\sigma, g^r\right) \end{array} \right] \right| \leq \frac{t}{s} + \frac{1}{n^c}$$

where $O^y{}_t(x) \stackrel{\triangle}{=} x^y$, but which is restricted in that the oracle can only be queried t times before it stops responding. Additionally, G represents a compact description of any information necessary to compute multiplications on the group elements, and not the elements of the group.

Observe that in the above assumption we have that when the adversary has no access to the oracle ($t = 0$), then the definition reverts to the traditional DDH assumption. We note that our restriction of a constant number of queries to the static oracle is because it suffices for our needs and is a weaker assumption than allowing a polynomial number of such queries. Still, we suspect that this is also a reasonable assumption. Also note that because of the random-reducibility of traditional DDH, the adversary has always had access to such an 'oracle' for random queries. The added benefit from this assumption is that the adversary can make queries relating to the g^{x_i} of interest. Finally, we assume this assumption holds in the traditional groups where DDH is assumed to hold.

6 High-level protocol description

The DPD protocol is built on top of two cryptographic primitives: $\binom{n}{1}$-OT and PAKE. Efficient protocols that implement these primitives are well known in the cryptographic community. We give brief and intuitive descriptions of these two essential elements.

$\binom{n}{1} - OT$: this is a primitive between a chooser and a sender, where a chooser learns one string from n possible strings of the sender. The security properties of OT ensure the following security properties:

The chooser learns only the string that it has chosen from the sender.

The sender does not learn which string the chooser chose.

PAKE: this is a primitive in which a client secretly exchanges a cryptographic key with a server with which it has previously shared a small (generally human-memorisable) password: the difference between the password and the cryptographic key begin that the former comes from a distribution with relatively low entropy. The security properties ensure that a passive adversary has essentially no chance of guessing the exchanged key, whereas an active adversary's ability to guess the exchanged key is no more than q/D, where q is the number of interactions that the adversary has with the client or server and D is the size of the dictionary from which the passwords are chosen. Observe that if the key-exchange is based on a password, this is essentially an optimal security condition, as an adversary can always guess a client's password with a probability of q/D. This is done by choosing q different passwords from the dictionary, and actively logging on to the server with each of them and seeing if one of them is correct.

A PAKE protocol gives us a secure method for key-exchange with a server, assuming the protocol is run correctly. However, it does not prevent phishing attacks because a phisher, using a passive doppelganger attack, will display on a user's machine a window that has the same user-interface as that of the PAKE protocol, and ask the user to enter her username and password. If she is tricked, and enters the information, then it is recorded by the phisher, as there is no security back-end to the fake window the phisher has displayed. This key example demonstrates the main attack our protocol must overcome. More generally, consider any protocol where the user enters her password completely and then receives feedback, and note that a phisher is likely to be able to duplicate the interface to such a protocol. Therefore, a phisher can generate a fake interface with no security back-end, and attempt to have users enter their passwords in the fake interface.

The goal of DPD is to only disclose the user's entire password at the point in which she is convinced she is talking to the intended site (and not a phisher). When she is convinced she is talking to the correct site she can be confident that the disclosure of her password will not cause her harm. Because of the step-by-step disclosure of the password, DPD essentially offers a graceful failure model for authentication.

The high level idea of DPD is the following. Imagine that the server has a database of easily distinguishable images. In particular, assume (for explanatory purposes only) that there are as many images as possible prefixes of passwords. For the sake of example, let's suppose the characters in a password are chosen from an alphabet of size 26. In DPD, when the user enters the first character in her password she is returned one of the first 26 images in the database. Specifically we think of the first character of her password as indexing into the image database via an OT protocol. When the user enters the second character of her password, she is returned one of the next $(26)^2$ images in the database which is indexed (via another OT protocol) by the first two characters of the password. This process continues for each character in the user's password. Once an image has been returned for each character of the user's password, and they are all the images she was expecting, then she can be relatively sure that she is interacting with the correct protocol, and therefore can initiate a traditional PAKE protocol with the server.

For a practical implementation, it would be excessively slow to transfer a large number of images via an OT protocol, even for the most efficient of OT protocols. Therefore, we imagine that instead of having a database of images, the server has a database of indexes to images that are actually transferred. The user then uses the index to look up the image in a local database, or the user can alternatively use the index to produce an image via random-art techniques (such as those presented in Dhamija (2000)).

7 DPD and its security

Ideally, we would like our DPD protocol to behave like a PAKE protocol, but with the added property that associate with each client's username and password is a vector of values $(y_1, .., y_c)$, which we can think of as representing images that the user expects to see after entering each character of her password. Similarly, we think of having each client's account on the server associated with the client's username and password as well as a random function $\mathcal{I}$. We would like to think of $\mathcal{I}$ being given to a trusted third party, and as the client enters her password, after each character ϕ_i the value of $\mathcal{I}(\phi_1, \ldots, \phi_j)$ is revealed to the client, and only if $\mathcal{I}(\phi_1, \ldots, \phi_j) = y_j$ for each j, will the user be convinced that she is dealing with the correct server and agree to perform the password authenticated protocol with it.

7.1 An intuitive description of the protocol

In order to create our protocol, one initial thought might be to consecutively perform a series of PAKE protocols, one for each character in the password $\phi = (\phi_1, ..., \phi_m)$ that the client has shared with the server. The 'shared password' for the ith execution of a PAKE protocol is the ith character, ϕ_i, of the password ϕ. If both parties agree on ϕ_i as the ith character of the password, then a shared key would be established between the client and server, and this can be used to transmit the image corresponding to the ith character of the password Φ from the server to the client. The security properties of the PAKE protocol would ensure

an adversary, masquerading as the server, would not learn the characters of the password, when sent by the client.

Unfortunately, such a proposal has several problems. Firstly, when the server and the client do not agree on a character in the password, then which image is displayed to the user? Since the client can easily establish the disagreement over characters, it might seem reasonable for the client to display a random image in such a scenario. However, this will not work as it would permit an adversary acting as a client to easily perform a server-probing attack on the client's password: the adversary would guess the first character of the password, and look at the image displayed. It would then repeat the protocol with the server, guessing the same first character. If the image displayed in both cases is the same, then with high probability this is the correct first character of the password, otherwise the adversary is guaranteed that this is not the first character of the password. By repeating this process, the first character can be determined, and then the process repeated. In order to prevent such a simple probing attack, we would like to ensure that for any incorrect prefix of the password the same incorrect image is displayed, no matter how many times the protocol is invoked.

An OT protocol solves many of the above problems. It allows one to have a database of images: one image for each possible prefix of the password. When the first character of the password is input by the the client or adversary, it is used to index into the database via the OT protocol to retrieve the corresponding image. Because the database is fixed, it does not matter how many times an adversary imitates a client, the same password prefix will always result in the same sequence of images. Further, because the OT protocol ensures that the server does not learn what position of the database was queried, if the adversary were simulating the server it would not learn the password prefix supplied by the user. Finally, because the server is guaranteed that the client learns exactly one database entry, there is no fear that a large number of feedbacks can be retrieved by a phisher with a small number of interactions with the server. If this were possible, then the phisher could use the collected feedback information to later perform an off-line doppelganger attack.

This OT solution seems quite reasonable and seemingly solves the problem. However, when one considers that the running time for even the fastest OT algorithms is, by necessity, proportional to the size of the database, and the database size would be roughly proportional to the space from which clients and servers select passwords, and it is clear this is not a practical implementation. In order to circumvent this problem, a series of OTs are performed on databases the size of the password alphabet (one OT for each character in the password), where each character of the password is used to index into a separate database. These databases are populated with 'image feedback' that is dependent on the previous selection of the user, thereby imitating the notion of performing an OT on the entire prefix of the password. However, this dependency contradicts the goal of keeping the user's selection secret from the server. Therefore, before the client performs an OT with the server, it needs to communicate to it the prefix of the password that has already been entered, so the server's database can be made dependent on it. This must be done in a blinded fashion, so that nothing about the prefix is learned by the server, but in a manner that allows the server to make the DB dependency. This is done through the use of some Diffie-Hellman exchanges.

Once the entire protocol has been entered by the client and it has received all of the expected feedback, then it should be convinced that it is talking to the correct server, but the server still has not received authentication from the client. Therefore, a traditional PAKE protocol is then executed. The entire protocol is described in the next section.

8 The DPD protocol

From the DDH assumption,[5] we assume that for a given security parameter n $\mathrm{IG}(1^n) \rightarrow (G, g, q)$ where g is the Generator of the group G and q is the order of the group. We will assume that these are fixed for the remainder of the protocol's description.

Let Σ be the alphabet over which passwords are chosen, where we assume that $|\Sigma|$ is a small but reasonable constant: in practice this might be in the range of 64–256. We assume that a client C and a server S have previously shared a password $\phi_C = (\phi_0, .., \phi_{m-1}) \in \Sigma^m$ where m is the length of the password in characters, in practice this would be approximately 8. When this password was agreed upon, the server randomly selected a Pseudo-Random Function (PRF) $F_C : \{0,1\}^n \rightarrow \{1, ..., q-1\}$ from an appropriate generator. This function is used to construct an alternative PRF $\mathcal{I}g, F_C : \bigcup_{i=1}^{m} \Sigma^i \rightarrow G$, which can be computed in an interactive and blinded way by the DPD protocol. The feedback for the user on input of ϕ_i is $\mathcal{I}_{g,F_C}(\phi_0, ..., \phi_m)$. The function $\mathcal{I}$ is defined below:

$$\mathcal{I}_{g,F_C}(\phi_0) \triangleq g^{F_C(\phi_0)} = y_0$$

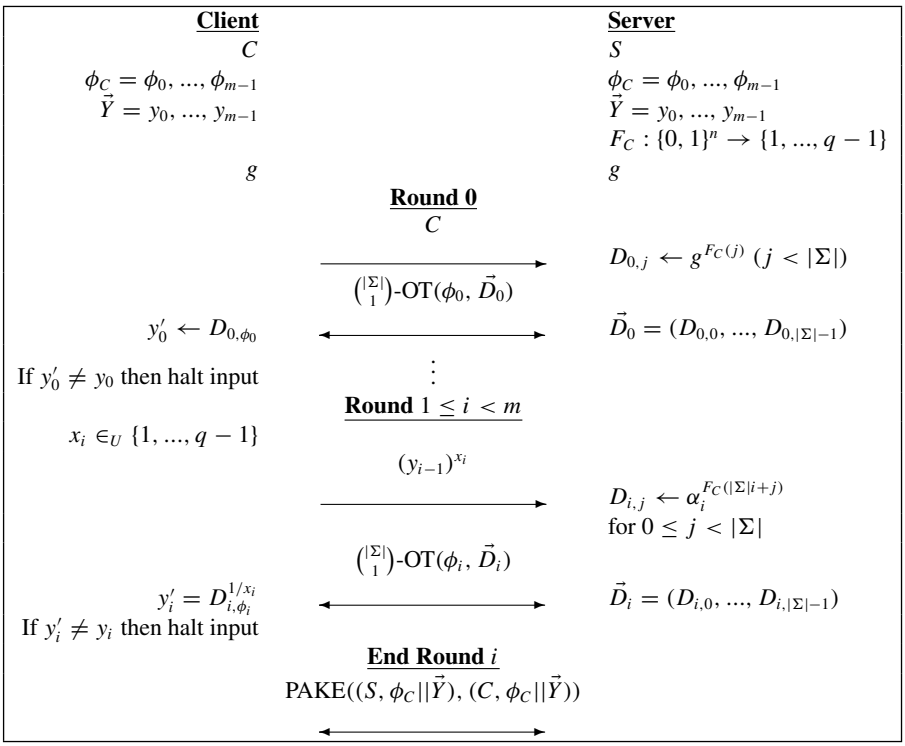
$$\mathcal{I}_{g,F_C}(\phi_0, ..., \phi_i) \triangleq y_{i-1}^{F_C(|\Sigma|\cdot i+\phi_i)} = y_i \quad (1 \leq i)$$

The DPD protocol is given in Figure 1, followed by the OT protocol intended to be used in Figure 2. Note that it is only the OT protocol that uses the random oracle H. This OT protocol is a simple modification of that presented by Naor and Pinkas (2001), which itself was a modification of a protocol presented by Bellare and Micali (1989). We refer the reader to these references for proofs of security of the OT protocol. We do note specify a particular PAKE protocol, as their efficiencies are fairly similar, and the protocol is only called once. Therefore, any of the PAKE protocols described in Katz et al. (2001), Bellovin and Merritt. (1992, 1993) and Gentry et al. (2005) should suffice.

8.1 Security against passive doppelganger attacks

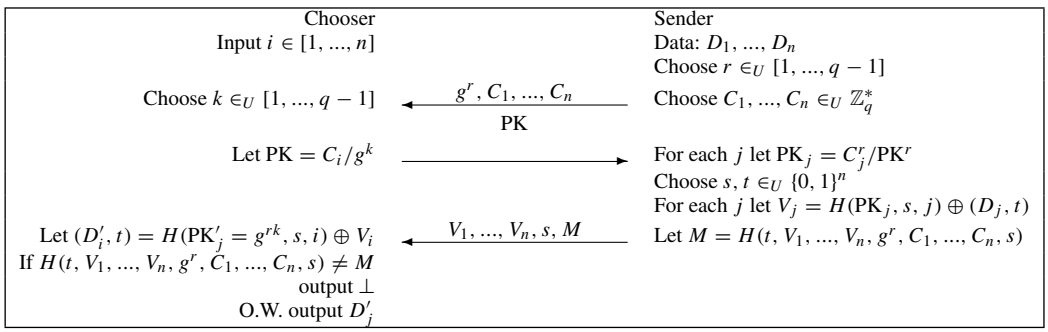
In order to prove security against passive doppelganger attacks, we consider a scenario where an adversary wishes to convince a specific user C to log-on to his fraudulent phishing website, that is spoofing a server S that the client has previously established a password ϕ with. We assume that the adversary has access to an honest server as an oracle. The adversary is allowed to interact with this server as many times as it wishes. Intuitively, it is at this point that the adversary attempts to learn all information that is needed

Figure 1 A depiction of a DPD protocol between the client C and the server S



Note: In the description $\binom{|\Sigma|}{1}$-OT$(\phi_i, \vec{D}_i)$ represents the execution of the oblivious transfer protocol described in Figure 2 where the client C is selecting the ϕ_ith element from the possible choices represented by $\vec{D}_i$, and where the previous image history (y_{i-1}) is carried along, blinded by the client/chooser using the blinding factor x_i. This allows the image of round i to be a function of all password characters *up to and including* the ith character, without requiring the OT function to allow an explicit selection from such a large corpus in each round. For reasons of efficiency, this is an important practical feature given that the OT protocol used has a linear complexity in the number of possible choices. Finally, PAKE$((S, \phi_C||\vec{Y}), (C, \phi_C||\vec{Y}))$ denotes the execution of the PAKE protocol by the client and server, where each use the password P_C concatenated with $\vec{Y}$ as the password for the PAKE protocol, and the usernames S and C as inputs to the protocol.

Figure 2 An efficient OT-protocol based on that of Naor and Pinkas (2001) that is provably secure in the RO-model



Note: As noted in Figure 1, the protocol has a linear complexity in the number of possible choices.

to fool the user later into logging on to the phishing site. Afterward, the adversary is given a challenge password ϕ, and asked to produce the images that correspond to it. This is to model the fact that a user should stop entering her password when incorrect feedback is returned. The adversary is successful if it can return $\mathcal{I}_C(\phi)$ (i.e., the appropriate feedback corresponding to the password ϕ)

We begin by observing that the PRF $\mathcal{I}_C$ is really pseudo-random.

Theorem 1: *Assuming F is a PRFG and the DDH assumption holds relative to the cyclic group G with generator g then $\mathcal{I}$ is a PRFG.*

Proof Sketch: We think of $\mathcal{I}$ as a number of functions $\mathcal{I} : \bigcup_{i=1}^{m} \Sigma^i \rightarrow G$ as being a series of functions $\{\mathcal{I}^i : \Sigma^i \rightarrow G\}_{i<n}$. The proof is done by induction replacing each H_i with a corresponding random function. First, however, by the assumed security of the PRFG F, we replaced F_C with a randomly chosen function $\widehat{F}$. Because the construction of $\mathcal{I}^1_{g,F_C}$ ensures that F_C is never queried on the same input twice, we see that the output of $\mathcal{I}^1$ is that of a random function. For the inductive hypothesis we assume that the distinguishing capability of the adversary is approximately the same when for $1 \leq j \leq i$ the function $\mathcal{I}_j$ has been replaced by a random function $\widehat{\mathcal{I}}^j$, as it is when

all of the appropriate $\mathcal{I}$ functions are used. For the inductive step, assume for contradiction that there is a large gap in the distinguishing probability in the case where for $1 \leq j \leq i+1$ the functions $\mathcal{I}^j$ have been replaced with random functions $\widehat{\mathcal{I}}^j$. This implies that there is an ability to distinguish the output $\mathcal{I}^{i+1}$ from that of a random function $\widehat{\mathcal{I}}^{i+1}$ on some distribution of inputs ϕ which can be found through hybridisation techniques. The output of such a function is $\mathcal{I}^{i+1}_{g,G,F_C}(\phi_0, ..., \phi_{i+1}) = \widehat{\mathcal{I}}^i(\phi_0, ..., \phi_i)^{\widehat{F}(|\Sigma|\cdot i+\phi_i)} = g^{r'\widehat{F}(|\Sigma|\cdot i+\phi_i)}$, where $\widehat{\mathcal{I}}^i(\phi_0, ..., \phi_i) = g^{r'}$, and by the DDH assumption this cannot be distinguished from a random output.

Next, we consider our adversary's ability to generate appropriate feedback for a random password.

Theorem 2: *Let F be a PRF generator. For all probabilistic polynomial time oracle adversaries A, for all $c > 0$ and for all sufficiently large n: let $IG(1^n) \rightarrow (G, g, q)$ be a group generated in which the DDH assumption holds, and the order of the generator q is prime; Let C be a client and S be a server that have previously shared a password chosen uniformly at random from a dictionary $\mathcal{D} = \Sigma^m$ where m is a small constant; assume the server has associated PRF F_C with the client, and established the feedback $\mathcal{I}_C(\phi) = y_1, ..., y_n$; then:*

$$\Pr\left[\begin{array}{c} F_C \in_U F \\ \phi' \in_U \mathcal{D} \\ A^S_C(1^n) \rightarrow \sigma;\ A(\sigma, \phi') \rightarrow \mathcal{I}_{g,C}(\phi) \end{array}\right] \leq q/|\mathcal{D}| + 1/n^c$$

where q represents the number of times A has commenced a new attempt to log-on with the server oracle S. The probabilistic experiment includes the selection of F_C, ϕ', the random choices made by A and random choices made by the oracle S_C.

Proof Sketch: In this scenario, it is clear that the PAKE protocol that is executed at the last step of the DPD protocol neither hinders nor helps the adversary, as this theorem is about learning the function $\mathcal{I}$. Imagine that the adversary is honest but curious. Then in this setting each interaction of the adversary with the server oracle S, giving a password ϕ' results in the adversary learning $\mathcal{I}_C(y)$ for each prefix y of ϕ'. The security properties of the OT-protocol ensures nothing more is learned. The pseudo-randomness of the function $\mathcal{I}_C$ guarantees that knowing the value of the function on certain values in the domain gives no predictive power for other points in the domain. Therefore, when A receives ϕ', the probability that the adversary has already queried $\mathcal{I}_C(\phi)$ is at most $q/|\mathcal{D}|$, and the probability that it can successfully produce $\mathcal{I}_C(\phi)$ without querying it is negligible.

Next, we must consider what happens when the adversary does not honestly follow the client's protocol. There are two places where this can occur: when the adversary sends a flow PK in an execution of an OT subprotocol and when the client sends the blinded α_i in each round $i > 0$, which allows the server to compute the function $\mathcal{I}_C$ appropriately.

Suppose the adversary ever sends an incorrect PK flow in one of the OT-Protocols, then the security properties of the OT-protocol (Naor and Pinkas, 2001) state that there is a simulator that the adversary could have executed, and received the same distribution on outputs of the OT, and therefore we can consider an adversary that never sends errant PK flows, as they could always be simulated. Next, assume that the adversary sends an incorrect value α'_i, as opposed to the value α_i that the honest client would compute. The protocol will clearly continue to function, but the values in the vector $\vec{D}_i$ change. In particular in this scenario the adversary essentially gains access to an oracle that will compute $v^{F_C(|\Sigma|i+j)}$ for a value $j \leq |\Sigma|$ of the adversary's choosing. Suppose that an adversary that makes such queries outputs $\mathcal{I}_{g,C}(\phi)$ with probability non-negligibly better than $q/|\mathcal{D}|$ when only q interactions with S are performed. This implies that having access to this ability to calculate the oracle is beneficial to the adversary, and contradicts the LSDDH assumption when at most $|\Sigma^m|$ queries to the oracle are allowed. We note that such a limit on oracle queries is sufficient, because an adversary who makes this many interactions with S can always recover the complete description of $\mathcal{I}$.

8.2 *Security against adversarial clients and servers*

We consider the models where we have an adversarial client interacting with an honest server and the opposite, an adversarial server interacting with an honest client. We show that in each case the adversaries are at least as restricted as they are in a PAKE protocol.

Theorem 3: *An adversarial client's ability to share a key with an honest server is no better than an adversary's ability to break the embedded PAKE protocol (i.e. no better than approximately $q/|\mathcal{D}|$, where q is the number of interactions of adversary with the honest server and $\mathcal{D}$ is the size of the dictionary).*

Proof Sketch: Since the adversary has no predisposition as to what the correct feedback should be, there is no advantage in receiving any of it. In particular, given a client adversary that shares a key with probability p with an honest server using the DPD protocol, we can construct an adversary that has the same effectiveness against the embedded PAKE protocol. The adversary would simply choose a random function $\mathcal{I}_C$ to simulate the interaction in the first rounds of the protocol until the PAKE protocol was initiated at the end of the DPD protocol, at this time, the adversary would interact with the legitimate PAKE protocol.

Since the simulation is essentially perfect, the adversary's success rate against the PAKE protocol would be no worse than its success against the DPD protocol.

Theorem 4: *An adversarial server's ability to share a key with an honest client is no more than $1/| < g > |$, where g is the generator of the group used in the DPD protocol.*

Proof Sketch: In order for an adversary to convince an honest client to share a password with it, it must provide the correct feedback. An adversary that has no information about the feedback is forced to guess randomly, as feedback is in effect chosen randomly. Since, the ability of the adversarial server to guess the appropriate feedback is

negligible (it must guess a randomly chosen group element just to get the feedback correct for the initial character of the password), the probability of adversaries success is no more than $1/|<g>|$.

8.3 Security against man-in-the-middle adversaries

Clearly, it is interesting and imperative to ask what security guarantees hold against strong MIM adversaries. In order to do this, a formal model must be defined. The authors are currently working toward a proof of security in a modification of the model proposed by Bellare et al. (2000) for proofs of security of PAKE protocols. Note that the traditional PAKE security models are not sufficient, as they do not allow for the interactivity allowed in the DPD model. Specifically, they do not allow for the security modelling of a user failing to continue in a protocol due to incorrect feedback being provided. One thing that should be evident is that in a proper model, the best a protocol can hope to achieve is that an adversary can completely determine a client's password with $\widetilde{\mathcal{O}}(m|\Sigma|)$ interactions with the server and the client (modelled as oracles by Bellare et al. (2000)) using the attack described on DPD in Section 4, but we believe this bound is achievable and likely to be met by our protocol. We also note that this is clearly a much more adversarial model than is needed for attacks performed by present day phishers, and therefore our security guarantees in the previous models have practical value.

9 Future work and conclusions

We believe that the phishing problem is currently one of the biggest threats to computer security, and more attempts must be made to apply different security techniques at the different choke-points of the phishing problem, in order for it to be addressed. We believe the DPD protocol is an interesting attempt to apply cryptographic techniques to that problem, and that the protocol developed is practical for situations where the servers computational load is not an issue. We hope this work encourages others to attempt to optimise this protocol or otherwise address the phishing problem with cryptographic techniques.

We also believe that the LSDDH assumption is worth further investigation, and hope it encourages others to consider its feasibility. We believe it will be a useful primitive in showing many interactive protocols based on Diffie-Hellman primitives are secure.

References

Anandpara, V., Dingman, A., Jakobsson, M., Liu, D. and Roinestad, H. (2007) 'Phishing IQ tests measure fear, not ability', *Usable Security (USEC)*.

Bellare, M., Boldyreva, A. and Palacio, A. (2004) 'An uninstantiable random-oracle-model scheme for a hybrid-encryptionproblem', in C. Canchin and J. Camenisch (Eds). *Advances in Cryptology-EUROCRYPT'04*, Springer.

Bellare, M. and Micali, S. (1989) 'Non-interactive oblivious transfer and applications', in G. Brassard (Ed). *Advancesin Cryptology–CRYPTO '89*, Volume 435 of *Lecture Notesin Computer Science*, Springer-Verlag, 20–24 August 1989, pp.547–557.

Bellare, M., Pointcheval, D. and Rog-away, P. (2000) 'Authenticated key exchange secure against dictionary attacks', *Lecture Notes in Computer Science*, Vol. 1807, pp.139–155.

Bellovin, S.M. and Merritt, M. (1992) 'Encrypted key exchange: password-based protocols secure against dictionary attacks', *Proceedings of the IEEE Symposium on Security and Privacy*, IEEE Press, May, pp.72–84.

Bellovin, S.M. and Merritt, M. (1993) 'Augmented encrypted key exchange: apassword-based protocol secure against dictionary attacks and password file compromise', *CCS'93: Proceedings of the 1st ACM Conference on Computer and Communications Security*, NewYork, NY, USA: ACM Press, pp.244–250.

Blum, M. (1983) 'How to exchange (secret) keys', *STOC '83: Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, New York, NY, USA: ACM Press, pp.440–447.

Brown, D.R.L. and Gallant, R.P. (2004) 'The static Diffie-Hellman problem', *Cryptology ePrint Archive, Report 2004/306*, Available at: http://eprint.iacr.org/.

Canetti, R., Goldreich, O. and Halevi, S. (1998) 'The random oracle methodology, revisited', *Proceedings of the 30th Annual Symposium on Theory Of Computing (STOC)*, Dallas, TX, USA: May, ACM Press, pp.209–218.

Chou, N., Ledesma, R., Teraguchi, Y., Boneh, D. and Mitchell, J.C. (2004) *Client-side Defense Against Web-based Identity Theft*, Proceedings of the Network and Distributed System Security Symposium 2004, The Internet Society.

Dhamija, R. (2000) 'Hash visualization in user authentication', *Proceedings of ACMCHI 2000 Conference on Human Factorsin Computing Systems*, Volume 2 of *Short Talks: Multimodal Interaction*, pp.279–280.

Dhamija, R. and Tygar, J.D. (2005) 'The battle against phishing: dynamic security skins', *Proceedings of the Symposium on Usable Privacy and Security (SOUPS) 2005*.

Emigh, A. (2005) 'Online identity theft: Technology, chokepoints and countermeasures', *DHS Report*.

Ford, W. and Kaliski Jr., B.S. (2000) 'Server-assisted generation of a strong secret from a password', *WETICE' 00: Proceedings of the 9th IEEE International Workshops on Enabling Technologies*, Washington, DC, USA: IEEE Computer Society, pp.176–180.

Franklin, M.K. and Reiter, M.K. (1997) 'Fair exchange with a semi-trusted third party (extended abstract)', *CCS'97: Proceedings of the 4th ACM Conference on Computer and Communications Security*, ACM, New York, NY, pp.1–5.

El Gamal, T. (1985) 'A public key cryptosystem and signature scheme based on discrete logarithms', *Proceedings of CRYPTO'84 on Advances in Cryptology*, New York, NY, USA: Springer-Verlag, New York, Inc. pp.10–18.

Garay, J.A., Jakobsson, M. and MacKenzie, P. (1999) 'Abuse-free optimistic contract signing', *Lecture Notes in Computer Science*, Vol. 1666, pp.449–466.

Gentry, C., Mackenzie, P. and Ramzan, Z. (2005) 'Password authenticated key exchange using hidden smooth subgroups', *CCS'05: Proceedings of the 12th ACM Conference on Computer and Communications Security*, New York, NY, USA: ACM Press, pp.299–309.

Goldwasser, S. and Kalai, Y.T. (2003) 'On the (in) security of the Fiat-Shamir paradigm', *Proceedings of the 44th Symposiumon Foundations of Computer Science (FOCS)*, IEEE Computer Society Press.

Griffith, V. and Jakobsson, M. (2005) 'Messin' with Texas: Deriving mother's maiden names using public records', *Applied Cryptography and Network Security (ACNS)*, Springer-Verlag.

Herzberg, A. and Gbara, A. (2004) *Technical Report 2004-23, Protecting (even) Naïve Web Users, or: Preventing Spoofing and Establishing Credentials of Web Sites, DIMACS*, October 30, Available at: http://dimacs.rutgers.edu/TechnicalReports/2004.html.

Jakobsson, M., Tsow, A., Shah, A., Blevis, E. and Lim, Y-K. (2007) What instills trust? a qualitative study of phishing.

Jakobsson, M. and Ratkiewicz, J. (2006) 'Designing ethical phishing experiments: a study of (ROT13) rOnl auction query features', *Proceedings of the 15th annual World Wide Web Conference*.

Katz, J., Ostrovsky, R. and Yung, M. (2001) 'Efficient password-authenticated key exchange using human-memorable passwords', in B. Pfitzmann (Ed). *Advances in Cryptology - EUROCRYPT' 2001*, Volume 2045 of *Lecture Notesin Computer Science*, Springer-Verlag, Berlin Germany, pp.473–492.

Kuo, C., Parno, B. and Perrig, A. (2006) 'Phool-proof phishing prevention', *Proceedings of Financial Cryptography and Data Security*.

MacKenzie, P., Patel, S. and Swami-nathan, R. (2000) 'Password-authenticated key exchange based on RSA', *Lecture Notes in Computer Science*, Vol. 1976, pp.599–613.

Naor, M. and Pinkas, B. (2001) 'Efficient oblivious transfer protocols', *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-01)*, January 7–9, 2001. New York: ACM Press, pp.448–457.

Schechter, S., Dhamija, R., Ozment, A. and Fischer, I. (2007) 'The emperor's new security indicators', *Proceedings of the IEEE Symposium on Security and Privacy*.

Smith, S.W. (2005) *Trusted Computing Platforms: Design and Applications*, Springer.

Soghoian, C. (2007) 'A deceit-augmented man in the middle attack against Bank of America's SiteKey service', April 10th 2007, Available at: http://paranoia.dubfire.net/2007/04/deceit-augmented-man-in-middle-attack.html.

Wu, M., Miller, R.C. and Garfinkel, S.L. (2006) 'Do security toolbars actually prevent phishing attacks?' *CHI' 06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA: ACM Press, pp.601–610.

Yahoo!Inc. (2006) Website, Available at: http://security.yahoo.com/article.html?aid=2006102507.

Notes

[1]We do not assume that the correct protocols are followed, but do assume the absence of keyboard loggers, and the like. A more detailed trust model is presented in Section 5.

[2]We note that such redirection attacks are not meaningful once a cryptographic key has been established between the target site and the user machine.

[3]This would otherwise contradict our requirement of allowing promiscuous accesses of target sites.

[4]As previously discussed, we will not prove such a property is necessary due to a lack of a formal MIM security model.

[5]We remind the reader that this is a special case of 5.5, and DDH is implied by it.