Delayed Password Disclosure

Markus Jakobsson
School of Informatics
Indiana University
Bloomington, IN, USA
markus@indiana.edu

Steven Myers
School of Informatics
Indiana University
Bloomington, IN, USA
samyers@indiana.edu

ABSTRACT

We present a new authentication protocol called Delayed Password Disclosure. 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 recognize an ongoing phishing attack, and prevent the disclosure of his/her entire password, providing graceful security degradation.

Categories and Subject Descriptors

K.6.5.4 [**Computing Milieux**]: Management of Computing and Information Systems—*Security and Protection, Authentication*

General Terms

Security, Algorithms, Theory

Keywords

Phishing, Password Authenticated Key Exchange, Oblivious Transfer, Doppelganger, Decisional & Static Diffie-Hellman Assumption, User Interfaces

1. INTRODUCTION

Apart from the direct damage phishing is doing to the financial industry, it is also seriously threatening to halt the expansion of ecommerce, 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 organizations 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 realize 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 [17, 22].

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

DIM'07, November 2, 2007, Fairfax, Virginia, USA.

(PAKE) [3, 19, 18], 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.

Within 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* – i.e., 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 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 of the user's machine, but may in principle also be used for any user authentication process. 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. That 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.

Our contributions.

We propose a protocol that permits a user interface that provides users with visual character-by-character feedback[2] as they enter their passwords, allowing users to stop entering their password if they obtain feedback that they do not recognize—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 recognize 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.

[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]While we discuss feedback throughout this paper as visual, it need not be.

2. RELATED WORK

The rapid rise of phishing attacks and their potential to have large negative effects on ecommerce has resulted in a significant number of researchers trying to solve the phishing problem. The approaches have varied widely, which is appropriate 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. [8] 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. 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 [16], which associates logos with the public keys of the certificates of visited sites. The use of the logos will hopefully create a better connection in the user's mind than the corresponding certificate.

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 path is available in [11]. A way to circumvent the traditional problem of trusted path was recently suggested by Parno, Perrig and Kuo [21]; their approach involved the use of an auxiliary device (a cell phone) to maintain trusted state. Dhamija and Tygar [10] approach the phishing problem by creating a trusted path between the user interface and the user. In their proposal a user must select a specific image which will be superimposed onto all dialog boxes presented by the browser; it acts as a shared secret between the user and the browser. This technique does not allow for promiscuous browser use, and is therefore incomparable to our threat model.

Passmark^{TM} (also branded as *SiteKey*) is a product of RSA Security (EMC), and has recently been deployed by the Bank Of America website to fight phishing. It augments on the traditional username/password interface by authenticating back to the user. It does so by first recognizing client browsers by means of previously installed cookies, and by requesting the user to enter his username; if the pair is recognized, then a user-specific image is displayed authenticating the site. Users are trained not to enter their password unless they recognize the image displayed to them. While Passmark is a similar approach to ours, the requirement of cookies requires non-promiscuous browsing, and thus is a solution for stronger model than the one we provide for. Ad-

ditionally, the client-side portion has some drawbacks. For example, cookies are not resilient to pharming, and their techniques do not provide any protection against such attacks. User studies by Schecter et al. [22] 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. However, these results are specific to interface deployed by Passmark, and we believe that an appropriately designed user interface can address this problem. We stress that our contribution is not interface specific, but rather a security protocol that allows for interfaces with feedback; these interfaces could be substantially different than that currently presented by Passmark.

3. DOPPELGANGER ATTACKS

While we have briefly mentioned doppelganger attacks, in this section we formalize the notion. In a *doppelganger attack*, an attacker controls one or more user interface devices (i.e., a display) on a victim'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 target site, while he or she instead is interacting with the attacker. We consider two classes of doppelganger attacks:

Offline 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 in a polynomial number of sessions in order to collect any pertinent information. Next, 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 not interface with the legitimate site during this time. If the credentials are released, the adversary is thought to be successful.

Let us now consider the need for interactivity to address offline doppelganger attacks in the absence of a trusted path between the user and the interface. Suppose to the contrary that the there is no feedback to the user *during* password entry. Any feedback given to the user *before* password entry can be duplicated by a offline doppelganger attack. 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, who will then enter her password. Alternatively, 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 may realize that she was attacked, but her password has been disclosed to the attacker.

Online Doppelganger Attacks.

In this case the attacker is permitted all of the benefits of an offline 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, as opposed to the protocol level that cryptographers are concerned with.

Given our model implies 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. In this attack model, without additional assumptions, the online doppelganger attack cannot be prevented.

4. A HIGH-LEVEL VIEW OF OUR GOALS

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 to maximize 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:

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 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, i.e., on all the characters that have been entered. This is both to maximize the min-entropy of the feedback, which in turn increases security, and to secure against temporary failures to recognize invalid visual feedbacks. Namely, and as we will later describe in more detail, we assume that users may occasionally fail to recognize 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 recognize invalid feedbacks are aggregated. However, this apparent contradiction can be resolved by the use of oblivious transfer techniques; these allow the client to index (using the credential prefix) a *database* of feedback items that is unique to the user in question.

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.

We provide a solution that allows for a careful analysis, while operating in a threat model that is practically oriented. We use oblivious transfer techniques to allow the exchange of feedback and credentials, without leaking the credentials. If the human user accepts the feedback items, then a traditional PAKE protocol is performed on the then-fully-entered credential. While the traditional use of oblivious transfer 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

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 pre-established shared secrets with target sites (passwords to be remembered and feedback to be recognized), and that these are relatively short (i.e., not of sufficient length to allow secure usage as cryptographic keying material.) There are no pre-established secrets between user machines and target sites.

Adversary.

We consider a remote 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 security contexts, as opposed to phishing which relies on a completely artificial security context provided by the browser's user interface.

The adversary is assumed to be able to direct selected victims from a target site to a site controlled by the adversary. 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). We consider a two-tiered adversarial model to capture two different possible strengths of the attacker. In a first model – which we refer to as the *offline* model – the adversary is relatively weak and may simply impersonate either client or server to the other, but is not able to able to perform an active man-in-the-middle attack.

In the second model – the *online* model – the adversary controls the scheduling of all network traffic, and may inject messages of choice in the network. While the online model is more commonly considered cryptographically, the offline 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 the bursty traffic patterns often associated with phishing attacks.[3] 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.

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[4]. Furthermore, we model the machine as a *semi-trusted* node: (a) 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; and (b) 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.

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. [5] 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 Oblivious Transfer 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. Further, many PAKE protocols make use of the Random Oracle model (the exception is the work of Katz et al. [18], but this protocol is not widely deployed). Finally, our protocol

[3]Given the risk of takedown, currently taking only in the range of a some hour using services like Cyota or Markmonitor, phishers are forced to perform attacks with a massive parallelism in a short period of time.

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

[5]While its known that arbitrary protocols proven secure in the random oracle model cannot be securely implemented [7, 1, 15], in practice it has shown to be an effective heuristics, and the RSA-OAEP sub-protocol 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.

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 scenario in which 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 ElGamal [13] in the case of chosen ciphertext attacks, and the Ford-Kaliski Key retrieval protocol [12]. This has been studied to some extent in the computational Diffie-Hellman case by Brown and Gallant [6], 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 adv ance. This is formalized below. We believe that this new assumption may of interest in its own right, and deserves further study.

Computational Assumption 1. Let IG be an algorithm that takes as input a unary security parameter, and outputs a cyclic 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:

$$\left| \Pr_{\substack{x_1,..x_s,y \in_{\mathcal{U}} Z_\ell \\ j \in_{\mathcal{U}} Z_s}} \left[\begin{array}{c} A^{O^{y}{}_t}(G, g, g^{x_1}, ..., g^{x_s}, g^y) \to \sigma; \\ A(\sigma, g^{(x_j y)}) \end{array} \right] - \right.$$

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

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 Decisional Diffie-Hellman 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. Finally, we assume this assumption holds in the traditional groups where DDH is assumed to hold.

6. CRYPTOGRAPHIC SUB-PROTOOCOLS

The DPD protocol is built on top of two cryptographic primitives: $\binom{n}{1}$-Oblivious Transfer (OT) and Password Authenticated Key Exchange (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:

1. The chooser learns only the string that it has chosen from the sender.
2. 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-memorizable) 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.

7. DELAYED PASSWORD DISCLOSURE AND ITS SECURITY

Ideally, we would like our DPD protocol to behave like a password authenticated key-exchange 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. [6] 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.

In order to create our protocol, one initial thought might be to consecutively perform a series of PAKE protocols, one

[6] For a practical implementation, it would be excessively slow to transfer a large number of images. Therefore, we imagine that instead of having a database of images, the server has a database of indexes to images that are actually transfered. 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 [9]).

for each character in the password $\phi = (\phi_1, ..., \phi_m)$ that the client has shared with the server. The 'shared password' for the i^{th} execution of a PAKE protocol is the i^{th} character, ϕ_i, of the password ϕ. If both parties agree on ϕ_i as the i^{th} 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 i^{th} 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. First, 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.[7]

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, 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 the nothing about the prefix is lear ned 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[8], we assume that for a given security parameter n $IG(1^n) \rightarrow (G, g, q)$ 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 to 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:

$$\begin{aligned} \mathcal{I}_{g,F_C}(\phi_0) &\triangleq g^{F_C(\phi_0)} = y_0 \\ \mathcal{I}_{g,F_C}(\phi_0, ..., \phi_i) &\triangleq y_{i-1}^{F_C(|\Sigma| \cdot i + \phi_i)} = y_i \quad (1 \leq i) \end{aligned}$$

The DPD protocol is given in Fig. 1, followed by the OT protocol intended to be used in Fig 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 [20], which itself was a modification of a protocol presented by Bellare and Micali [2]. 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 [18, 4, 5, 14] should suffice.

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

[7]If such learning were possible, then conceivably the phisher could later use the collected feedback information from a small number of interactions to perform an offline doppelganger attack.

[8]We remind the reader that this is a special case of 1, and DDH is implied by it

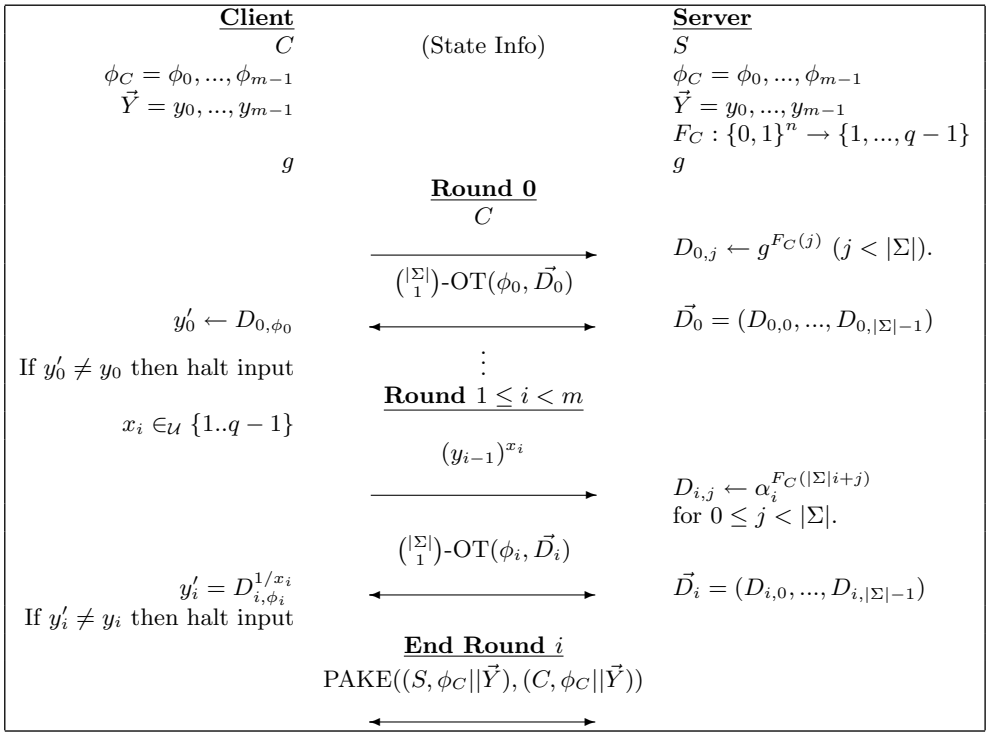

Figure 1: The DPD protocol between the client C and the server S. In the description $\binom{|\Sigma|}{1}$-OT$(\phi_i, \vec{D}_i)$ represents the execution of the oblivious transfer protocol described in Fig 2 where the client C is selecting the ϕ_i*th* element from the possible choices represented by $\vec{D}_i$. 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.

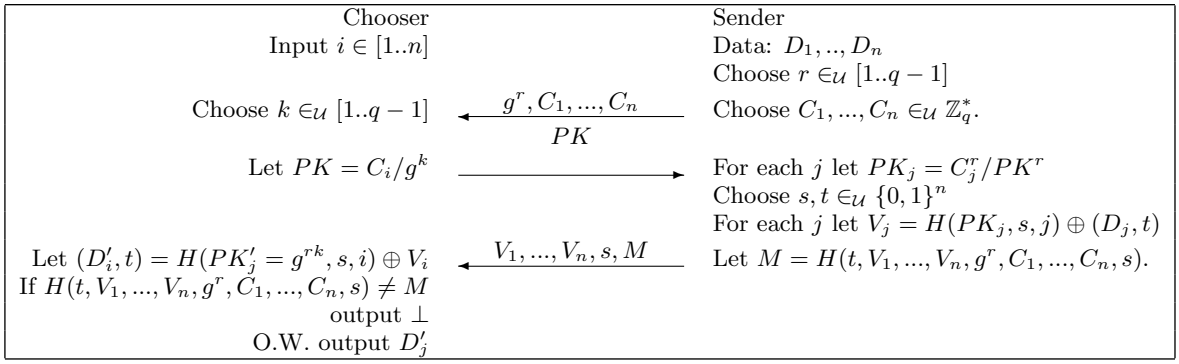

Figure 2: An efficient OT-protocol based on that of Naor and Pinkas [20] that is provably secure in the RO-model. As noted in Figure 1, the protocol has a linear complexity in the number of possible choices.

phishing web-site, 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 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. See Appendix A for a proof sketch. □

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

THEOREM 2. *Let F be a pseudo-random function 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:*

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

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. See Appendix A for a proof sketch. □

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 $\frac{q}{|\mathcal{D}|}$, where q is the number of interactions of adversary with the honest server and $\mathcal{D}$ is the size of the dictionary).*

PROOF. See Appendix A for a proof sketch. □

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. See Appendix A for a proof sketch. □

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 the model proposed by Bellare et al. [3] for proofs of security of PAKE protocols. In this model it is clear that the adversary can completely determine a client's password with $\widetilde{\mathcal{O}}(m|\Sigma|)$ interactions with the server and the client (modeled as oracles by [3]) using the described on DPD in Section 4, and we believe this is the best the attacker can do. We note that this is clearly a much more adversarial model than present day phishers are performing, 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 addresses. 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 other to attempt to optimize this protocol or otherwise address the phishing problem with cryptographic techniques.

10. REFERENCES

[1] M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In C. Canchin and J. Camenisch, editors, *Advances in Cryptology-EUROCRYPT'04*, pages 171–188. Springer, 2004.

[2] M. Bellare and S. Micali. Non-interactive oblivious transfer and applications. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 547–557. Springer-Verlag, 1990, 20–24 Aug. 1989.

[3] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. *EUROCRYPT-Lecture Notes in Computer Science*, 1807:139–155, 2000.

[4] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Press, May 1992.

[5] S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise. In *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*, pages 244–250, New York, NY, USA, 1993. ACM Press.

[6] D. R. L. Brown and R. P. Gallant. The static Diffie-Hellman problem. Cryptology ePrint Archive, Report 2004/306, 2004. `http://eprint.iacr.org/`.

[7] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proceedings of the 30th Annual Symposium on Theory Of Computing*

(STOC), pages 209–218, Dallas, TX, USA, May 1998. ACM Press.

[8] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell. Client-side defense against web-based identity theft, Apr. 2004.

[9] R. Dhamija. Hash visualization in user authentication. In *Proceedings of ACM CHI 2000 Conference on Human Factors in Computing Systems*, volume 2 of *Short talks: multimodal interaction*, pages 279–280. ACM Press, 2000.

[10] R. Dhamija and J. Tygar. The battle against phishing: Dynamic security skins. In *SOUPS 05:Proceedings of the Symposium on Usable Privacy and Security*, pages 77–88, New York, NY, USA, 2005. ACM Press.

[11] A. Emigh. Online identity theft: Technology, chokepoints and countermeasures. In *DHS Report*, 2005.

[12] W. Ford and J. Burton S. Kaliski. Server-assisted generation of a strong secret from a password. In *WETICE '00: Proceedings of the 9th IEEE International Workshops on Enabling Technologies*, pages 176–180, Washington, DC, USA, 2000. IEEE Computer Society.

[13] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[14] C. Gentry, P. Mackenzie, and Z. Ramzan. Password authenticated key exchange using hidden smooth subgroups. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 299–309, New York, NY, USA, 2005. ACM Press.

[15] S. Goldwasser and Y. T. Kalai. On the (in)security of the Fiat-Shamir paradigm. In *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS)*, pages 92–101. IEEE Computer Society Press, 2003.

[16] A. Herzberg and A. Gbara. Trustbar: Protecting (even naive). web users from spoofing and phishing attacks, 2004.

[17] M. Jakobsson and J. Ratkiewicz. Designing ethical phishing experiments: A study of (ROT13) rOnl auction query features. In *Proceedings of the 15th annual World Wide Web Conference*, pages 513–522, 2006.

[18] J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT' 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 473–492, Innsbruck, Austria, 2001. Springer-Verlag, Berlin Germany.

[19] P. MacKenzie, S. Patel, and R. Swaminathan. Password-authenticated key exchange based on RSA. *ASIACRYPT '00: Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security (Lecture Notes in Computer Science)*, 1976:599–613, 2000.

[20] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-01)*, pages 448–457, New York, Jan. 7–9 2001. ACM Press.

[21] B. Parno, C. Kuo, and A. Perrig. Phoolproof phishing prevention. In G. D. Crescenzo and A. Rubin, editors, *Financial Cryptography*, volume 4107 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2006.

[22] S. Schechter, R. Dhamija, A. Ozment, and I. Fischer. The emperor's new security indicators. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 51–65, 2007.

APPENDIX

A. PROOF SKETCHES

In this appendix we give the proof sketches for the previously stated theorems from Section 8.

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 replace 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 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 replace 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 hybridization 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. □

THEOREM 2. *Let F be a pseudo-random function 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:*

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

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 sub-protocol 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 ([20]) 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 LS-DDH 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 $\mathcal{I}$. □

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 $\frac{q}{|\mathcal{D}|}$, 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 $1/| < g > |$. □