ACM SIGACT News Distributed Computing Column 27

Sergio Rajsbaum*

Abstract

The Distributed Computing Column covers the theory of systems that are composed of a number of interacting computing elements. These include problems of communication and networking, databases, distributed shared memory, multiprocessor architectures, operating systems, verification, Internet, and the Web. This issue consists of:

- "Delayed Password Disclosure," by Markus Jakobsson and Steven Myers.

Many thanks to them for their contribution to this issue.

Request for Collaborations: Please send me any suggestions for material I should be including in this column, including news and communications, open problems, and authors willing to write a guest column or to review an event related to theory of distributed computing.

Delayed Password Disclosure

Markus Jakobsson and Steven Myers
School of Informatics
Indiana University at Bloomington

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.

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

*Instituto de Matemáticas, Universidad Nacional Autónoma de México. Ciudad Universitaria, Mexico City, D.F. 04510. rajsbaum@math.unam.mx

1 Introduction

Phishing is a form of online fraud that is doing increasing damage to the financial industry. In its traditional form, a victim is sent a fraudulent email message directing the victim to fraudulent website, normally hosted on a hacked machine, with some urgent call to action. The purpose of the call to action is typically to incite users to follow a link, by suggesting that they will receive some form of reward for following the link, suffer a penalty for failing to follow, or some combination of the two. The website that the victims are directed to is designed to mimic the appearance of a legitimate site, such as an online bank, vendor or payment system. The goal here is to continue the confidence game, initiated in the call to action, so that the victim remains convinced that she is interacting with the legitimate site. The fraudulent website will request from the vicim a number of credentials and other information of interest, such as personally identifying information. The goal of the attacker, termed a *phisher*, is to trick the victim into releasing his credentials to the fraudulent website, so that later they may be used to perform some form of fraud. In general, the credentials of interest are credit or banking related, the most common examples of which include credit-card numbers, or pairings of usernames and passwords for online banking sites. Phishers try to obtain as much information as they can from their victims. Phishers generally sell off the information that they collect to other individuals, called *cashers*, who specialize in monetizing the information, and the more information that a phisher can acquire on a given victim the more value it will fetch, as this information aids the cashers in their goal.

Apart from the social aspects of phishing, there are also technical components. These are often aimed at circumventing phishing countermeasures (see, e.g., [32, 37]) or spam filters (see, e.g. [35]), or at mining user-specific data of relevance to an attack (see, e.g., [26, 29, 31].) Thus, phishing is a complex threat, consisting of both a technical component and a social/psychological component. Its very complexity is also what makes it so difficult to defend against. As an example of why it is difficult to defend against, we can recall how security experts, until quite recently, argued that once the public became aware of the threat, then technical security features such as SSL would eradicate the problem. This is not happening. One reason is that while most people understand that there is an association between security and the presence of an SSL lock, many do not know how to interpret security indicators (see, e.g., [22, 14]) and few know that phishers can also use SSL [39] or simply add locks in the *content portion* of a webpage [33]. Second, it is known that there is a tremendous discrepancy between what typical users *know* and what they *practice*. An example of this is illustrated by recent studies involving eyeball tracking [46], in which it was concluded that most users rarely look for SSL indicators, much less choose to interact with these. While educating users is *necessary* to some degree, it is also difficult [43]. Moreover, it is not sufficient to explain the problems to the target audience, but one must also change their behavior. It is hard to untangle the technical aspect from the social aspect when we address the problem of phishing, and an understanding of both may be necessary to make significant progress.

1.1 Why Phishing Works

Phishing works due to a confluence of several different factors. It begins with a victim receiving a call to action that is essentially nothing more than a confidence game. People have been tricked by confidence games for as long as there have been con-men. While spam filters and other anti-spam technology can help reduce the exposure to these calls to action, it can easily be argued that even if spam were essentially eliminated, phishers would be able to con people into visiting their mimicked sites through other means. Once the intended victim visits the mimicked site, the confidence game is continued by providing a site that looks and has similar user experience to a legitimate site. This helps to establish or maintain a sense of trust with the fraudulent site, meaning that the victim is likely to provide it with information that he otherwise would not. Constructing a site that has a similar look and feel is, of course, not a difficult process, and can

even be automated [45]. Because the user believes he is interacting with the legitimate site as opposed to a fraudulent one, he does not worry about releasing potentially confidential information to it, as the user would often willing and safely release the same information to the corresponding legitimate institution.

However, there exist cryptographic mechanisms built into web-browsers that are supposed to aid users in determining the identities of the websites they are visiting. In particular, websites can use the Secure Socket Layer (SSL) protocol or its IETF standardized version, the Transport Layer Security (TLS) protocol, to push non-forgeable cryptographic certificates issued by trusted authorities to browsers who visit their sites, thereby authenticating themselves. Since many, if not most, legitimate commerce and banking sites use these certificates, it is reasonable to ask why consumers do recognize the mimicked sites by their lack of legitimizing certificate issued by a trusted certificate. There are varying reasons that can be given for this, but, almost universally, they can be grouped in to the category of usability issues: Users generally have little understanding of what or where the browser's most basic security indicators are, nevermind how to use them in order to determine the identity—or lack thereof—of the website with which they are communicating. For example, most users do not seem to be able to interpret the lock icon displayed by browsers to indicate an SSL/TLS session has been established. Typical users know to an even lesser extent that it may be clicked to display a certificate containing information about the purported identity of the website, along with information on the identity of the authority which is vouching for its identity.[1] The result is that when users visit phishing sites, the lack of appropriate cryptographic authentication is not noticed, and users tend to rely on indicators that they have come to trust in the non-digital world, such as the presence of brand logos and other look-and-feel branding intangibles, regardless of how inapplicable these indicators are as authenticators in the digital realm. A further, or more detailed explanation of the general phishing problem is beyond the scope of the current presentation, but interested readers are directed to [28] for comprehensive coverage of the issue.

The threat of phishing is real and expensive. 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. In fact, largely motivated by phishing scams, the Federal Deposit Insurance Corporation (FDIC) and the Federal Financial Insitutions Examination Council (FFIEC) mandated that all online banks in the US would need to substantially improve their authentication technologies by the end of 2006, beyond that of simple passwords over SSL/TLS established authenticated and encrypted channels [17].

1.2 Other Approaches

While punitive legislative approaches (see [11] for a review) 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. 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, or attacks on routers [44]), 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.

[1] In this discussion we will ignore another problematic issue: Certificates are often issued with weak or improper validation of the requester's identity.

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 [30, 41].

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 [4, 36, 34], 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. A incarnation of this attack is the so-called "picture in picture" attack [27].

In this presentation 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[2] terminals, some of which may not 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 one currently using the machine). Thus, such a setting shifts the emphasis to that of protecting against malware (as is needed 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 user 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.

[2]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.

Our proposal. 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 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 password authenticated key exchange (PAKE), is proven secure in several critical ways based on 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 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. [12] 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 [25], 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 organizations certifying that a site can be trusted and the trusted site. Further, it is postulated that 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 [16]. A way to circumvent the traditional problem of trusted path was recently suggested by Parno, Perrig and Kuo [40]; their approach involved the use of an auxiliary device (a cell phone) to maintain trusted state. Dhamija and Tygar [15] 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 dialog 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 dialog box. For this

solution to be used, the user must have a pre-established secret (the user's image) with the browser, and so this technique does not allow for promiscuous browser use, the model we address.

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 [42] for a comprehensive discussion). While trusted paths are typically concerned with how to secure data input (e.g., by standardized 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 directly 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 webpages 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 recognizing client machines by means of previously saved cookies, and by requesting the user to enter his username. If the user and his machine are recognized 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 recognize the image displayed to them. The product also has a server-side component that analyzes login attempts and usage patterns, looking for fraudulent activity.

While Passmark is a promising approach, the client-side portion has some drawbacks. First, 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. Second, given that cookies are not resilient to pharming (see, e.g., [28]), 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 other independent means. User studies by Schecter et al. [41] 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., [1]) to introduce a bias. However, a similar observation was made in the context of the "eBay user greeting" in [30], where subjects were not aware of participating in a study, much less a phishing study.

A similar technique to Passmark is that of Yahoo!'s 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 [48], but instead of identifying the *user* (which requires asking for the user name), it only recognizes 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 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 presentation. 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 is necessary for cookies.

3 Doppelganger Attacks

While we have briefly mentioned doppelganger attacks, in this section we formalize 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. Note that the attacker needs not present an exact duplicate or functionality of a given target site, but rather it must only be convincing enough to confuse average users: Images and interfaces that are clearly incorrect to experts are often mistaken as legitimate to average users, due to their more casual understanding of security indicators, browser chrome and standardized interfaces. We will 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 a polynomial number of sessions in order to learn the behavior 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), but the attacker is not permitted 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 *offline* doppelganger attacks. 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 —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 realize that she was attacked and this may be beneficial, but her password has been disclosed to the attacker at this point.

Online Doppelganger Attacks. In this model 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 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). Note that

even if a proposed security protocol is secure against an adversary that has complete control of the network the point is irrelevant as in a doppelganger attack, the protocol is not being executed.

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 offline 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 herein would no longer exist. While it may appear that the problem of concurrent signature exchange (e.g, [7, 19, 21]) 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 maximize the entropy of the distribution from which secret credentials are drawn, by not processing and providing feedback for password entry 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, 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, 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 by making it more likely that a phisher cannot duplicate correct feedback without knowledge of the password; 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. 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.

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, such as temporarily suspending the account; 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 $q = 26 + \log_2 26$ interactions with the server and client and the attack is guaranteed to learn the first character of the password. This could then naturally generalized to retrieve the remaining characters of the password. All in all, this would result in an attack that uses only mq 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 $qm/(26^m)$. 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 off offline 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 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 password-based key exchange 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, 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 (MITM) 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 insufficient, 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. . Finally, we note that such a MITM adversary is substantially more powerful than those currently enjoying success with phishing attacks, We also note that while the online MITM model is more commonly considered in much of the literature, 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 odd network traffic patterns often associated with MITM attacks that are actually deployed in practice; 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[3], 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).

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.

User assumptions. It is important to model users in a realistic manner, as their behavioral 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 [47]. They are better at noticing the presence of incorrect information [33] than the absence of correct information [30], 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.

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 [10, 2, 24], in practice it has shown to be an effective heuristic, 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. 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 attacker. Further, many PAKE protocols make use of the Random Oracle model (the exception is the work of Katz et al. [34], 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 the Static Diffie-Hellman (SDH) problem, where the adversary has limited access to an exponentiating oracle. In the traditional DDH assumption, informally, an adversary is given a generator g of a cyclic group, and g^x and g^y for random x and y, and then a value z that is either a random group element or g^{xy}; the assumption states the adversary cannot distinguish between the two possible values for z. This is opposed to the original computational Diffie-Hellman assumption which assumes that an

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

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

adversary, when given g^x and g^y cannot compute g^{xy} . The SDH assumption is a strengthening of the computational DH assumption that addresses the security issue of protocols that effectively give adversaries access to static DH oracles $O^y(w) \stackrel{\triangle}{=} w^y$ that raise group elements w to the power y, when only g^y is known by the adversary. Examples of protocols that rely on this security include ElGamal [20] in the case of chosen ciphertext attack security, and the Ford-Kaliski Key retrieval protocol [18]. 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. The SDH assumption has been studied to some extent in the computational Diffie-Hellman case by Brown and Gallant [8], 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.

6 High-Level Protocol Description

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 being 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.

The goal of Delayed Password Disclosure 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. Again for the sake of example, let's suppose the characters in a password are chosen from an alphabet of size 26. In delayed password disclosure, 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 oblivious-transfer 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 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 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 [13]).

7 The Delayed Password Disclosure Protocol

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 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. If this were possible, then the phisher could use the collected feedback information to later perform an offline doppelganger attack.

This OT solution seems quite reasonable and seemingly solves the problem. However, it is clear this is not a practical implementation when one considers that the running time for even the fastest OT algorithms is, by necessity, proportional to the size of the sender's database, and the database size is proportional to the users' password space. 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 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.

7.2 A Detailed Description

Using the DDH assumption, we take a cyclic group G of order q where the assumption is assumed to hold, find a generator g. 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 in the range of 8 to 64. 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 $\mathcal{I}g, F_C : \bigcup_{i=1}^{m} \Sigma^i \rightarrow G$, that maps from the domain of password-prefixes to group G. This function is used as it 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:

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

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 [38], which itself was a modification of a protocol presented by Bellare and Micali [3]. We refer the reader to these references for proofs of security of the OT protocol. We do not 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 [34, 5, 6, 23] should suffice.

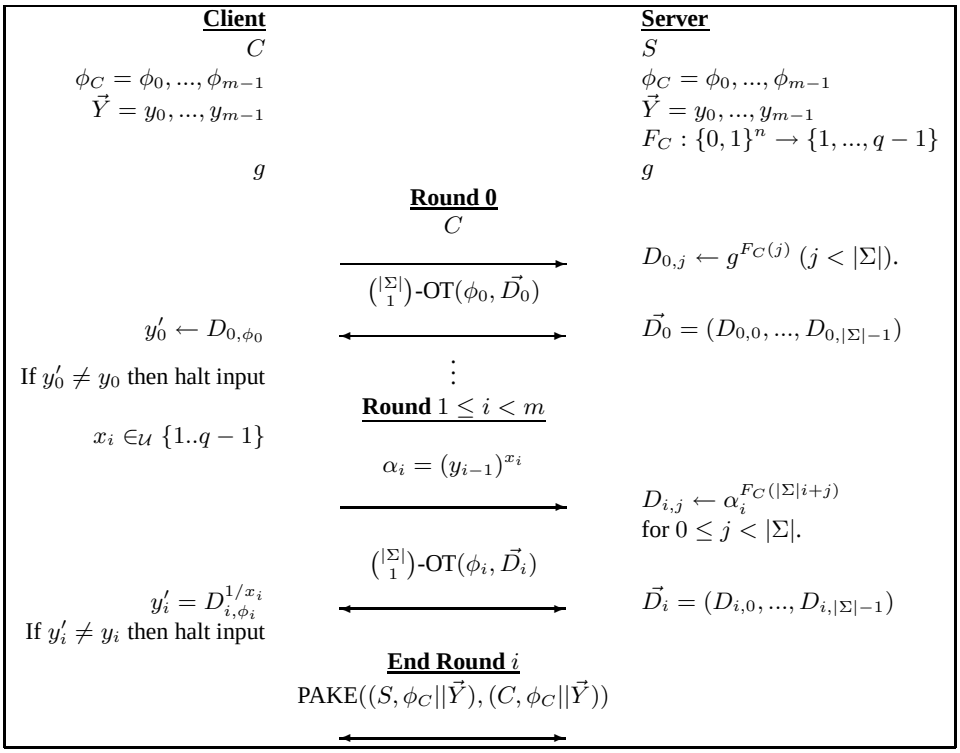

Figure 1: A depiction of a DPD protocol between the client C and the server S. In the description $\binom{|\Sigma|}{1}$-OT$(\phi_i, \vec{D}_i)$ represents the execution of the oblivious transfer protocol described in Fig 2 where the client C is selecting the ϕ_i*th* element from the possible choices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

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 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,

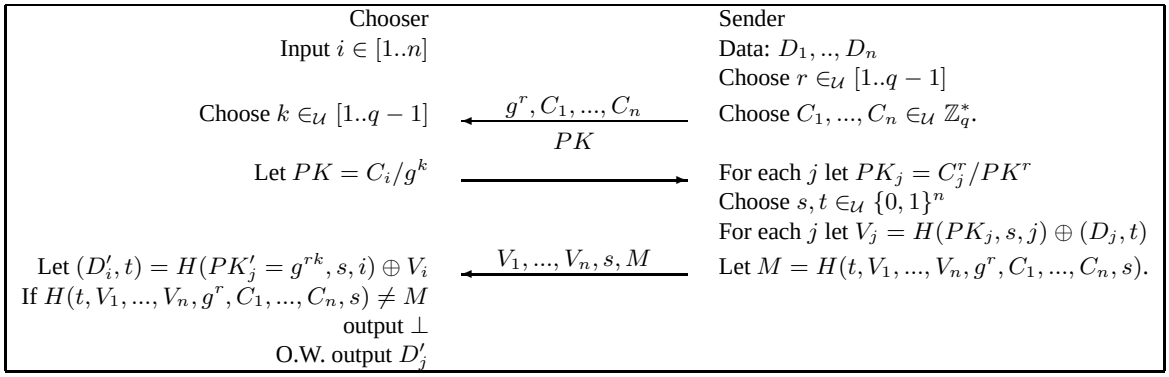

Figure 2: An efficient OT-protocol based on that of Naor and Pinkas [38] that is provably secure in the RO-model. As noted in Figure 1, the protocol has a linear complexity in the number of possible choices.

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).

If the adversary interacts as an honest party with the server, that is it runs the correct cryptographic protocol, while trying to determine the sequence of images corresponding to a user's password, then it clearly can learn only a sequence of images per password prefix login attempt. The pseudo-randomness of $\mathcal{I}$ ensures that given the feedback for a given password prefix, feedback cannot be predicted for other prefixes. The security properties of the OT protocol ensure that only one sequence of feedbacks is retrieved by the adversary per interaction with the server.

Cryptographic Security Security against a passive eavesdropper follows directly from the security guarantees of the OT and PAKE protocols, which guarantee security against such adversaries, and the fact the remaining network flows consist of public or random data, which the adversary could have produced itself, and thus does not contain valuable information.

Next, consider an active adversary who tries to impersonate a legitimate server, when interacting with a legitimate client. In this case the OT and PAKE security protocols ensure that the adversary does not learn the client's password if it does not both the password, and the correct feedback the server is supposed to provide. There are two reasons for this, the OT protects the prefix inputs of the user and the PAKE protocol ensures minimal information is lost in the case where the client and adversarial server do not share the same password. Further, because the DPD protocol concatenates the feedback strings to the DPD password for use in the PAKE protocol, this ensures that not only must the adversary know the correct password, but the correct feedback strings.

In the case of an adversarial client interacting with a legitimate server, then an attempt to interact with the server by feeding it a password results in the adversary learning nothing more than the whether or not the attempted password is valid, as well as the feedbacks that correspond to the attempted password. That no information is leaked on feedback that does not correspond to the selected password is guaranteed by a combination of the OT protocol's security guarantees and the strengthened SDH assumption. The strengthened assumption is needed here, for if in some round i the adversary does not compute $\alpha_i = y_{i-1}^{x_i}$, but rather sends some other value $\widehat{\alpha_i}$, then the honest server will still raise it to appropriate exponents derived from F_C, and these values are returned in the following rounds through the OT protocol. Yet, our computational assumption guarantees that in such cases, the adversary learns no information.

Finally, one should consider a strong adversarial model in which the adversary controls the entire network, and can have multiple authentication sessions interact concurrently. In such situations one needs strong cryptographic models such as those proposed by Bellare et al. [4] or Canetti [9]. Security in such models is desirable as the Internet, in principal, allows adversaries to manipulate traffic in a large range of fashions. While the works previously mentioned provide basic frameworks, they must be extended in order to allow such a proof. We are currently pursuing such work. However, as previously alluded to, these models allow attacks that are sufficiently stronger than those seen in practice, and further, currently many strong attacks that would be practical in this model are MITM attacks that are currently largely detectable through other mechanisms that are in place, such as the detection of odd or anomalous network traffic metrics.

References

[1] Vivek Anandpara, Andrew Dingman, Markus Jakobsson, Debin Liu, and Heather Roinestad. Phishing IQ tests measure fear, not ability. In *Usable Security (USEC)*, 2007. `http://www.informatics.indiana.edu/markus/papers/phish6.pdf`.

[2] 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.

[3] Mihir Bellare and Silvio 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 August 1989.

[4] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. *EUROCRYPT-Lecture Notes in Computer Science*, 1807:139–155, 2000.

[5] Steven M. Bellovin and Michael 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.

[6] Steven M. Bellovin and Michael 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.

[7] Manuel Blum. How to exchange (secret) keys. In *STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 440–447, New York, NY, USA, 1983. ACM Press.

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

[9] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In Bob Werner, editor, *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS-01)*, pages 136–147, Los Alamitos, CA, October 14–17 2001. IEEE Computer Society.

[10] Ran Canetti, Oded Goldreich, and Shai 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.

[11] Fred Cate. Liability for phishing (chapter 18), In Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft. Editors Markus Jakobsson and Steven Myers, 2006.

[12] Neil Chou, Robert Ledesma, Yuka Teraguchi, Dan Boneh, and John C. Mitchell. Client-side defense against web-based identity theft, April 2004.

[13] Rachna 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.

[14] Rachna Dhamija, J. D. Tygar, and Marti Hearst. Why phishing works. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI2006)*, 2006.

[15] Rachna Dhamija and J.D. 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.

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

[17] Federal Financial Institutions Examination Council. Authentication in an internet banking environment, October 2005. `http://www.ffiec.gov/pdf/authentication_guidance.pdf`.

[18] Warwick Ford and Jr. 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.

[19] Matthew Franklin and Michael K. Reiter. Fair exchange with a semi-trusted third party, 4th ACM Conference on Computer and Communications Security. pages 1–5, 1997.

[20] Taher El 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.

[21] Juan A. Garay, Markus Jakobsson, and Philip MacKenzie. Abuse-free optimistic contract signing. *Crypto '99*, pages 449–466, 1999.

[22] Simson Garfinkel and Robert Miller. Johnny 2: a user test of key continuity management with S/MIME and Outlook Express. In *SOUPS 05:Proceedings of the Symposium on Usable Privacy and Security*, pages 13 – 24, New York, NY, USA, 2005. ACM Press.

[23] Craig Gentry, Philip Mackenzie, and Zulfikar 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.

[24] Shafi Goldwasser and Yael Tauman 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.

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

[26] Collin Jackson, Andrew Bortz, Dan Boneh, and John C. Mitchell. Protecting browser state from web privacy attacks. In *Proceedings of the 15th annual World Wide Web Conference*, pages 737 – 744, 2006.

[27] Collin Jackson, Dan Simon, Desney Tan, and Adam Barth. An evaluation of extended validation and picture-in-picture phishing attacks, In Usable Security 2007. `http://www.usablesecurity.org/papers/jackson.pdf`.

[28] Markus Jakobsson and Steven Myers (Eds). Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft, 2006.

[29] Markus Jakobsson, Tom N. Jagatic, and Sid Stamm. Phishing for clues: Inferring context using cascading style sheets and browser history, 2005. `http://www.browser-recon.info`.

[30] Markus Jakobsson and Jacob 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.

[31] Markus Jakobsson and Sid Stamm. Invasive browser sniffing and countermeasures. In *Proceedings of the 15th annual World Wide Web Conference*, pages 523 – 532, 2006.

[32] Markus Jakobsson and Alex Tsow. Making takedown difficult (chapter 11), In Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft. Editors Markus Jakobsson and Steven Myers, 2006.

[33] Markus Jakobsson, Alex Tsow, Ankur Shah, Eli Blevis, and Yung-Kyung Lim. What instills trust? a qualitative study of phishing, 2007. Usable Security '07, `http://www.informatics.indiana.edu/markus/papers/trust_USEC.pdf`.

[34] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Birgit 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.

[35] Changwei Liu. Fighting obfuscated spam, Second APWG eCrime Researchers Summit, October, 2007.

[36] Philip MacKenzie, Sarvar Patel, and Ram 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.

[37] Tyler Moore and Richard Clayton. Examining the impact of website take-down on phishing, Second APWG eCrime Researchers Summit, October, 2007.

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

[39] Netcraft News. More than 450 phishing attacks used SSL in 2005. http://news.netcraft.com/archives/2005/12/28/more_than_450_phishing_attacks_used_ssl_in_2005.html.

[40] Bryan Parno, Cynthia Kuo, and Adrian Perrig. Phoolproof phishing prevention. In Giovanni Di Crescenzo and Avi Rubin, editors, *Financial Cryptography*, volume 4107 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2006.

[41] Stuart Schechter, Rachna Dhamija, Andy Ozment, and Ian Fischer. The emperor's new security indicators. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 51–65, 2007.

[42] Sean W. Smith. *Trusted Computing Platforms: Design and Applications*. Springer, 2005.

[43] Sukamol Srikwan and Markus Jakobsson. Using cartoons to teach internet security. DIMACS Technical Report 2007-11, July, 2007. `http://www.informatics.indiana.edu/markus/documents/security-education.pdf`.

[44] Sid Stamm, Zulfikar Ramzan, and Markus Jakobsson. Drive-by pharming, 2006. Indiana University Technical Report TR641, `http://www.symantec.com/avcenter/reference/Driveby_Pharming.pdf`.

[45] Webwhacker 5.0. `http://www.bluesquirrel.com/products/webwhacker/`, accessed July 26, 2007.

[46] T. Whalen and K.M. Inkpen. Gathering evidence: use of visual security cues in web browsers. In *Proceedings of the 2005 Conference on Graphics interface*, pages 137–144, 2005.

[47] Min Wu, Robert C. Miller, and Simson L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 601–610, New York, NY, USA, 2006. ACM Press.

[48] Yahoo! `http://security.yahoo.com/article.html?aid=2006102507`, accessed July 26, 2007.