COMBATING PHISHING THROUGH ZERO-KNOWLEDGE AUTHENTICATION

by

PAUL KNICKERBOCKER

A THESIS

Presented to the Department of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Master of Science

June 2008

"Combating Phishing Through Zero-Knowledge Authentication," a thesis prepared by Paul Knickerbocker in partial fulfillment of the requirements for the Master of Science degree in the Department of Computer and Information Science. This thesis has been approved and accepted by:

Dr. Jun Li, Chair of the Examining Committee

Date

Commitee in charge:	Dr. Jun Li, Chair
	Dr. Ginnie Lo
	Dr. Reza Rejaie

Accepted by:

Dean of the Graduate School

An Abstract of the Thesis of

Paul Knickerbocker for the degree of Master of Science

in the Department of Computer and Information Science

to be taken June 2008

Title: COMBATING PHISHING THROUGH ZERO-KNOWLEDGE AUTHENTICATION

Approved: ____

Dr. Jun Li

Phishing is a type of Internet fraud that uses deceptive websites to trick users into revealing sensitive information. Despite the availability of numerous tools designed to detect phishing, it remains a steadily growing threat. The failure of current anti-phishing solutions is largely due to their focus on detecting phishing rather than addressing phishing's root cause: insecure web authentication.

Using a combination of the zero-knowledge mechanism and two-factor authentication I present ZeKo, an authentication mechanism that is immune from phishing attacks, cryptanalysis and man-in-the-middle attacks. ZeKo takes into account the psychological behavior of users and remains secure even when the user is deceived. The proposed system not only prevents phishing attacks but also has considerable benefits over traditional authentication mechanisms, making it well suited for a wide range of applications.

CURRICULUM VITAE

NAME OF AUTHOR: Paul Knickerbocker

PLACE OF BIRTH: Kalamazoo, MI

DATE OF BIRTH: May 3, 1980

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene

DEGREES AWARDED:

Master of Science, University of Oregon, 2008
Bachelor of Science, University of Oregon, 2004

AREAS OF SPECIAL INTEREST:

Network security, Anonymous trust relationships, Anti-Phishing, Web 2.0 Security, Internet worm detection and modeling

PROFESSIONAL EXPERIENCE:

Information Services Support, Whittier Wood Products, 2001-2003
Network Administrator, FOOD for Lane county, 2003-2006
Research Assistant, University of Oregon, 2007-2008

GRANTS, AWARDS AND HONORS:

1st place, Graduate Research Poster Contest, 2007: *Bringing End-to-End Cryptography to Web Mashups*
2nd place, Graduate Research Poster Contest, 2007: *A Preliminary Scheme for Combating Phishing with Zero Knowledge Authentication*
1st place, Graduate Research Poster Contest, 2005: *On the Design and Analysis of Characteristics-Based Worm Detection Heuristics*
Geoffery Eric Wright Outstanding Junior Award, 2004

PUBLICATIONS:

Shad Stafford, Jun Li, Toby Ehrenkranz, and Paul Knickerbocker, *GLOWS: A high-fidelity worm simulator* , Tech. Rep. CIS-TR-2006-11, University of Oregon, 2006.

Jun Li and Paul Knickerbocker, *Functional similarities between computer worms and biological pathogens*, Computers & Security, vol. 26, no. 4, pp. 338-347, June 2007.

ACKNOWLEDGMENTS

I wish to thank Dr. Jun Li for his support of my research during my graduate and undergraduate studies. I also wish to thank Dr. Du Li at Nokia Research for his support and input during the exploration of this topic, as well as Dr. Ginnie Lo for her academic advising during my graduate studies. Partial support for this research was provided by NSF Award CNS-0644434 and NSF Award IIS-0133871. Portions of this text were originally written for an NSF grant proposal on anti-phishing authentication.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

The Phishing Epidemic

Phishing is a form of electronic identity theft in which *phishers* use deceptive websites to trick users into disclosing sensitive information. Phishers lure potential victims, often via e-mail, to websites designed to mimic the appearance of legitimate sites where the victims would have an existing account or would be looking to do business. Under the impression that the website is legitimate, the victim fills out the forms on the site with personal information and transmits it back to the phisher. Phishing websites usually seek to obtain credit card information or website usernames and passwords that can be used to purchase goods or transfer funds.

Phishing activity has steadily increased to match the growth of web commerce, recently taking on epidemic proportions. Symantec's reporting system research saw 1,088 unique phishing e-mails per day for the first half of 2007, an 18% increase over the last six months of 2006 [2]. The Anti-Phishing Working Group confirmed 31,709 unique phishing sites in June 2007 alone [3]. In 2006, the response rate to phishing e-mails were estimated to be as high as 19%, with up to 5% of all recipients revealing sensitive information [4]. Of those that revealed information to phishers, 45% reported their information was used to perform unauthorized purchases or transfers [5]. Another study in 2007 on password habits estimated that at least 0.4% of all users studied had reveled usernames and passwords to phishers [6].

The financial services industry has borne the brunt of the phishing epidemic; the cost of phishing activity to US banks and credit card institutions estimated at $1.2 billion in 2003 [7], $1.7 billion in 2006 [8] and $3.2 billion in 2007 [9]. For the first half of 2007, phishing attacks targeting the login ID's for financial service websites accounted for 79% of all reported phishing attacks [2]. Locally, phishing attacks have targeted the Lane county based Oregon Community Credit Union [10] and even the University of Oregon's Duckweb system [11].

Although the impact of phishing has been substantial, it is just one part of the growing problem of identity theft in a digital age. The rise of web commerce and web-accessible services has expanded the amount of personal information held by agencies and on-line businesses, making unintended disclosure more probable. In 2006, 15 million Americans became victims of identity theft; the average cost to the victim was $1,244, with only 54% of the stolen funds ever being recovered [12]. To compound the problem of identity theft via phishing, many phishers also sell or give away the stolen information on the Internet's IRC black markets, leading to account abuse by multiple identity thieves [13].

One of the most disturbing trends in recent phishing activity is its use as a source of income for criminal organizations. Well organized and financed phishing operations have been linked to both organized crime and terrorism [14, 15].

The ZeKo Solution

Using a combination of the zero-knowledge mechanism and two-factor authentication I present ZeKo, an authentication mechanism that is immune from phishing attacks, cryptanalysis and man-in-the-middle attacks. Zero-knowledge systems provide leak-proof communications between parties and cryptographic security while two-factor systems prevent the inadvertent disclosure of authentication elements by the user, the unique combination of these authentication mechanisms allows ZeKo to compensate for the psychological behavior of users and remains secure even when the user is deceived. The proposed system not only prevents phishing attacks but also has considerable benefits over traditional authentication mechanisms, making it well suited for a wide range of applications.

In this work I will first review the current state of anti-phishing systems and their flaws in Chapter II. I will then present the design and function of my own anti-phising authentication system, ZeKo, in Chapter III along with its related work. The performance of ZeKo is evaluated against other authentication systems with similar properties in Chapter IV and I conclude with a review of the extended areas of research in Chapter V.

Background on Phishing

User deception, while one of the least sophisticated forms of attack, is also one of the most successful. Compared to the complexity of circumventing the defenses of a securely designed system through technological means, deceiving the user is often the quickest and most reliable methods of gaining access. The technique of bypassing security by deceiving users is often called *social engineering* and is a classic tool used by computer hackers.

Phishing is a type social engineering attack that exploits the user's ignorance about how websites and security technology works. Increasing numbers of people integrate technology and the Internet into their lives with very little understanding about how the underlying systems work or what security threats they are vulnerable to. Users often determine the legitimacy of a website based on the website content alone [16] and carry over decision strategies from real-world interactions that are inappropriate for digital interactions[17]. Phishing uses people's natural inclination to use the appearance of a website as a primary indication of validity rather then the technical indicators such as SSL certificates.

Phishing's predecessors, false-login trojans, exploited the assumption of legitimacy by the user before there was even an Internet to phish. False-login trojans harvested passwords using programs that created input prompts that appeared very similar to the legitimate login prompts, or by replacing the system code for the legitimate login with their own versions. In addition to collecting passwords for later retrieval by the attacker, the trojans also commonly installed back-door access for the attacker through the use of a "super" password that was always accepted [18]. The introduction of Ctrl-Alt-Del to the Windows login procedure was in direct response to the threat posed by login trojans (Ctrl-Alt-Del being a "Secure Attention Sequence" accessible only to the OS) [19]. False-login trojans continue to operate today to through the use of rootkits and have expanded to infect virtual machines [20].

On systems not connected to a network, the collection of login credentials had only limited value; physical access was still required to retrieve the password records and gain access to the system. The construction of a fake login also took considerable technical knowhow, involving knowledge of the inner-workings of compilers and the targeted operating system. These restrictions on the usefulness of fake logins hampered their growth. Unlike the level of skill required to construct false-login trojans, phishing can be performed with only a rudimentary understanding of HTML. In fact, construction of phishing websites is often handled by phishing kits [2] and requires very little technological expertise.

Phishing on attacks are commonly initiated using e-mail. The phisher builds a website that mirrors the look of a legitimate website they wish to gain access to, then the phisher sends out e-mails with forged “from” addresses that make the e-mail appear to be from the legitimate website. The phishing e-mail is made to look official (through the use of logos and “business” language) and compels the victim to follow a hyperlink embedded in the message. A common phishing e-mail is one which tells the victim that the website has lost their information and needs them to re-enter it; the messages also commonly suggest that their account will be terminated if the information is not re-entered. Sometimes phishing attacks are even self-referential, requesting information on the pretense being part of the website’s anti-phishing system.

The link in the e-mail that leads the user to the phishing site is often misleading, with the displayed text of the hyperlink giving the URL of the legitimate site while the link itself takes the user to the phishers site. The URLs of phishing sites are also commonly worded to appear related to the targeted website, such as http://PayPal.phisher.com/ and http://www.pay-pal-IDs.com/ which both appear to be related to PayPal to those not familiar with fully qualified domain name (FQDN) system. Other phishing links simply use IP addresses (e.g., http://64.32.34.23/index.html) to avoid DNS registration at all.

A more subtle form of phishing is called *pharming*, it involves implanting fraudulent DNS records to provide users with IP addresses to phishing websites when they request the IP addresses of legitimate sites. Pharming is usually accomplished by poising the cache on a compromised DNS server, or by re-configuring a victim's DNS server list to point to a DNS server set up by the pharmer. Re-configuring the victim's DNS list can be done using viruses, trojan horses or by compromising the network's DHCP server (like in drive-by pharming, which exploits unconfigured wireless routers [21]).

Recently, there have also been a rise in *vishing*, where the personal information is collected over the phone and not through a fraudulent website. Similar to standard phishing, vishing uses mass e-mails to compel victims to call a phone number and give their account information to an operator which is in fact an identity thief. The rise in vishing may be related to the expansion of IP telephony, providing voice lines which can be set up an run with minimal exposure for the visher. Given the press attention surrounding phising in the last few years, users may be more comfortable disclosing account information over the phone and find the use of a phone more legitimate.

CHAPTER II

CURRENT ANTI-PHISHING APPROACHES

Current Systems

The current anti-phishing systems that have seen significant deployments on the Internet can be classified into three groups: (i) password management tools, (ii) heuristic-based phishing detection mechanisms, and (iii) website credential systems. Most anti-phishing systems usually incorporate strategies from several of these groups to counterbalance each group's relative weaknesses. The most popular tools use a combination of phishing detection and website credentials to alert the user to potential phishing threats during web browsing. Increasingly these tools have become an integral part of the web browser [22, 23].

While current schemes provide a level of protection against phishing sites, they do not address the fundamental flaws in Internet authentication or provide any protection in the event of a successful phishing attack.

Password Management

One of the simplest mechanisms for preventing phishing involves automating the password entry mechanism through the use of password managers [24, 25]. Users expose their passwords by submitting them to websites that appear legitimate; password managers prevent accidental exposures by making the determination of the website's identity based solely on technical means. Password managers take the user out of the password submission process and so limit the possibility of submission to an unauthorized server. Managers have the added advantage of keeping track of a wide range of web passwords, reducing forgotten password events and helping enforce strong, unique passwords.

Most phishing sites can be easily distinguished from the legitimate sites they mimic by examination of the URL; but most web users are unfamiliar with URL structure and so rarely notice the discrepancies [26]. Password managers provide control on the release of passwords by managing their password records based on the FQDNs of the servers the password was bound to. While phishing URLs can be constructed to appear to related to the mimicked site, the mechanical matchup between the FQDN associated with a password in a password management system is immune to these deceptions.

Obfuscation is another technique used by password managers to prevent unintended password release, commonly through the use of cryptographic hashes [27, 28]. Password obfuscators use the combined hash of a user-selected password and a piece of data unique to the legitimate site, often the DNS domain name, as the password to the site. When the user authenticates, the obfuscators recreates the hash based on the users password and the data from the site being submitted to. In the event the site being submitted to is a phishing site the password submitted will not be the password to the legitimate site because the seed from the website used in creating the password is different from the legitimate site, producing a different result.

Despite the advantages of password managers, the problems that come along with them outweigh the benefits. Even under perfect conditions most password managers are susceptible to pharming attacks, due to their assumptions about the validity of DNS entries. Portability becomes a problem as users come to rely on the managers to remember their passwords; portability is even more of a problem with obfuscators where the password is unknown to the user and the software is required to generate the website's password. Ultimately, password managers are a stop-gap measure against the flaws inherent in the username/password authentication mechanism itself.

Heuristic-based Phishing Detection

Heuristic-based phishing detection schemes attempt to identify phishing by looking for phishing characteristics in a website's HTML or in the text of e-mail messages. Most heuristics are simplistic and check for common phishing tactics such as misleading URLs or hyperlinks to IP addresses. Systems such as Spoofguard [29] and numerous browser plug-ins [30, 22, 23, 31] use machine-readable properties of a website and compare them with the websites in the browsers history to determine whether a site is legitimate (previously visited sites assumed to be more legitimate). Similar heuristic-based schemes are used identify phishing e-mails that lead users to fraudulent sites through deceptive hyperlinks [32, 33]. Contextual analysis [34] and comparisons of the structure of rendered webpages to pre-established signatures [35, 36] are some other, more advanced, detection techniques proposed.

A recent survey tested the top 11 heuristic-based phishing detection toolbars and found that the best performing system, Spoofguard [29], could detect up to 95% of phishing websites, but with a false positive rate of 42%. No other toolbar was able to detect more than 87% of the phishing websites analyzed [37]. An evaluation of advanced machine-learning detection techniques have shown similar results [38]. Another study tested users' responses to warnings from the toolbars — despite correct identification of the phishing site by the toolbars, a third of the participants still revealed usernames and passwords to fraudulent sites [26].

Even at its best, heuristic-based phishing detection can only stop a limited amount of phishing and is almost entirely helpless against pharming do to the heuristic's assumption of DNS integrity. The heuristics used to detect phishing behavior also require constant revision as phishing techniques change. For every phishing trick that a heuristic learns to detect, there is soon another one that avoids detection. The incomplete protection of phishing detection makes it primarily a defensive mechanism and can not be expected to make a significant impact against future phishing attacks.

Website Identification

Because phishing is fundamentally the result of misidentifying websites, many anti-phishing systems are build around establishing positive website identification before the user enters their authentication credentials. These solutions either use technological means (such as certificates) which positively identifies the site, or they present the site to the user in a manner which cannot be duplicated.

Cueing

One of the most popular identification mechanisms adopted by websites to prevent phishing is the use *cues* to signal their legitimacy to the user. The largest implementations are those using visual cues. In simple visual cuing [39, 40, 41], the user has some unique image that the legitimate website would display but a phishing site would not be able to. More sophisticated systems make cues integral to the usage of websites, like through the use of random visual keypads for password entry [42]. Other systems use complete page transformations at the browser level to signal a page's legitimacy [43].

Despite the popularity of cue systems, studies have shown that cue systems are ineffective at stopping phishing attacks [16, 26, 17, 44]. While some cues may be ignored by users due to poor design, most are dismissed simply due to the user's lack of understanding about website security. One study found that 97% of users of a website with a prominent visual cue system would continue to enter their passwords despite the cue being completely absent [44].

Verified Credentials

A more technical approach to website identification is based on website credentials that automatically identifying legitimate sites from phishing sites, usually through the use of a trusted third party. This approach aims to enable legitimate websites to prove their own identities in a way phishing sites could not mimic, thereby detecting phishing sites by their absence of credentials. Public Key Infrastructure (PKI) is the most widespread example of a credential system to identify legitimate websites. When a user establishes a secure connection with a website, typically through the Secure Socket Layer (SSL) protocol, the user can verify the public key certificate of the website in order to make sure the website is legitimate.

In a perfect world, credentials would effectively eliminate phishing. Most users, however, do not understand how a website is verified using certificates or understand the connection between encrypted communications and website security. One study found that while 80% of users associated the image of a padlock on a website with security, only 40% knew about the SSL indicator built into the web browser (also a padlock) that signaled a secure connection with a valid certificate [17]. Another study found that only 23% of users used the web browser's SSL indicator as a factor in deciding whether or not a website was legitimate, and only 9% of users had ever examined certificates [16].

Restriction Lists

The most common method of website identification utilized by most popular anti-phishing toolbars is the use of whitelists and blacklists [45, 30, 46, 22, 47, 23, 31]. Blacklists are usually populated through reports of phishing activity; the lag before a phishing site is detected and added to a blacklist limits the usefulness of blacklists against rapidly emerging threats. Studies have shown that blacklists from Google and Microsoft only list 55-65% of the active phishing sites reported in another well-known blacklist [48].

Whitelists have all the same problems as blacklists in terms of incompleteness, as well as some of their own. For a whitelist to determine if a site is legitimate, it first has to determine "who" the website claims to be. The problem of determining "who" a website is claiming to be is just as hard a problem as determining if a given site is legitimate in the first place. Both blacklists and whitelists also usually assume the validity of DNS information, making both of them vulnerable to pharming.

Flaws in Current Approach

All of the currently proposed systems seek to combat phishing by stopping the user from entering phishing information into a fraudulent website. Given a perfect scenario where all phishing sites could be detected and users always followed the tool's advice, this approach would be sufficient. In reality, some users can always be assumed to make bad decisions and detection can always be assumed to be less then perfect. Combatting phishing through detection and prevention is undermined by the psychology of the user. For a solution to be effective it must not only provide security when the user follows good security behavior, but also when they do not.

All the current anti-phishing tools suffer from two major design flaws: (1) the reliance on user vigilance, and (2) the focus on detection rather than prevention.

The Reliance on User Vigilance

The steady increase of phishing attacks over time despite the introduction of anti-phishing technology strongly suggests that current solutions for combating phishing are incapable of dealing with the problem. Rather than building strong account security into the authentication process, websites often rely on the user's vigilance in protecting their login information. This approach only works if users are sophisticated and able to effectively detect phishing sites—an assumption which has been repeatedly shown to be incorrect [16, 26, 17, 44, 49]. Security indicators built into browsers that can indicate phishing are usually ignored [44, 16], as are the security toolbars designed to detect phishing activity and alert users [26].

The most common response to the problem is user education, but even using the most effective teaching methods available average users are still fooled by 30% of the phishing attacks [50]. A joint US and Canadian governmental working group on mass marketing fraud [51] stated in very direct terms the problem with relying on user education to solve the problem:

> "Although consumer education programs are an important component of the fight against phishing ... they will not suffice to provide adequate protection for the public as phishers continue to refine their attack techniques."

Any *realistic* solution for preventing phishing must not depend on assumptions about user behavior; instead, the solution should compensate for user behavior that is susceptible to phishing.

Detection without Prevention

The other fundamental deficiency of current anti-phishing solutions is the focus on detection rather then prevention. Anti-phishing solutions which are designed to simply detect phishing attacks will always have a level of accuracy below 100%. Making them not a solution, but simply a damage control measure. Risk mitigation schemes are at best sandbag solutions and do nothing to address root causes of the security problem. Even when phishing detection systems can have detection rates as high as 91% [37], there are still phishing attacks which slip by. The failure rate for an anti-phishing system, no matter how small, has very real economic and human consequences.

Due to their false positive rates, most of the phishing detection schemes do not restrict insecure behavior and still rely on the user to make sound decisions on website security. Even in studies where anti-phishing tools had an 100% detection rate, users distrusted the tool's conclusions and still entered sensitive information, reducing the effective protection rate to between 50-77% [26].

Phishing detection schemes also commonly assume the integrity of the DNS information, making them susceptible to pharming [37].

A *complete* solution to the phishing problem will not try to simply reduce the threat of phishing, but will instead seek to eliminate it through solidly designed authentication.

An Effective Anti-Phishing Strategy

My research moves in a fundamentally different direction from current anti-phishing work by acknowledging on the following essential facts:

- *Users will disclose their login credentials to unauthorized third parties.* Users determine whether or not a website is trustworthy based on psychological factors, factors that can be manipulated into giving the appearance of legitimacy to fraudulent sites [52]. The average user will ignore tell-tale signs of phishing attacks and even the toolbars designed to warn them of phishing sites [26].

- *Phishing sites fool even sophisticated users* [16, 49] Increasing user awareness about the threat posed by phishing does not necessarily lead to better security practices by the user [17, 44]. Phishing, as it exists today, will continue to occur in the future until the attack no longer provides the phisher with access to the user's account.

- *The responsibility for the security of a system rests with the system designer, not the user.* Designers cannot rely on the user's compliance to a set of best practices in order to protect the integrity of the user's account. Security for the user should be inherent in the system, rather than a result of the user's familiarity with security protocols. Systems which follow solid security design principles must take into account the psychological component that influences user behavior [53] and compensate for its flaws. Phishing is primarily the result of a flaw in the authentication mechanism that allows users to be psychologically manipulated.

- *The network is adversarial.* The username/password paradigm is an authentication scheme held over from the days when system access was done through physical terminals. It was assumed that physical terminals would always be legitimate and all that was needed was a simple mechanism based on a shared secret. With authentication over the Internet, the problem of untrustworthy logins becomes acute. Secure design demands that an open network like the Internet has to be considered to be adversarial, where attackers can be in control of the links and nodes through which the authentication traffic may pass. The authentication process should make no assumptions about the legitimacy of any party involved.

These realities are at the heart of my design philosophy, and drive my approach to the phishing problem. Rather than just trying to reduce the damage caused by phishing, I seek to remove the flaws from authentication design which allows phishing to be successful. I aim to provide an effective solution for all known phishing and third party attacks while fulfilling the technical specifications set out by the Internet Engineering Task Force for phishing-resistant web authentication [54].

CHAPTER III

ZEKO AUTHENTICATION

Design Overview

The goal of the proposed system presented in this paper is to provide a *realistic and complete* solution to the problem of password phishing on the Internet. The system compensates for insecure user behavior and anticipates the eventuality of of data disclosure. In order to achieve that end the system is resistant to both standard technical attacks on the authentication process as well as attacks via phycological deception.

The proposed system, which is called *ZeKo* (in reference to zero-knowledge authentication), seeks to take the burden of security off the individual users through secure design. ZeKo prevents phishing through a process that reveals no sensitive information during authentication and which uses credentials that cannot be stolen by phishers. The system also makes no assumptions about the integrity and secrecy of the authentication data on the server, or any data transmitted over the network. The distrust and assumption of hostility by all elements on the network allows the mechanism to function securely in an openly adversarial environment.

In order to meet the goal of providing secure, phising-resistant authentication the system must meet the following requirments:

- Authentication that is resilient against the theft of authentication data, including a username and password disclosed through phishing.
- Authentication that is immune to man-in-the-middle attacks.
- Authentication that is resistant to cryptanalysis on intercepted communications or stolen authentication data.

ZeKo meets these requirements by using a unique mix of zero-knowledge and two-factor authentication.

Zero Knowledge Authentication

Zero-Knowledge authentication is a practical application of the concept of a *zero-knowledge proof.* In a zero-knowledge proof, one party can confirm whether or not a statement is true without reveling any other property on the statement [55]. This type of proof is especially useful in the context of remote authentication because of the risk of disclosure through the insecure transmission medium. If two parties agree to certain constraints beforehand, each party can use the truth of the other's statement as a form of positive identification without transmitting anything that can be used by a malicious party to impersonate either of the parties.

While zero-knowledge proofs can be constructed out of a wide range of problems (most notably all *NP* problems [56]), most of them are not practical for realistic authentication systems. Real-world zero-knowledge systems generally leverage the difficulty in solving the equation $v = g^x \bmod n$ for x. Zero-knowledge has found numerous applications in cryptography, being the basis for the Diffie-Hellman key exchange [57] and asymmetric cryptography (such as RSA [58]).

Along with cryptography, authentication is a natural application for zero-knowledge proofs, as the proof of statements can be used to confirm the identities of the participating parties. It is especially appealing in an environment such as the Internet where any disclosed information could be used by an attacker. In a zero-knowledge authentication system the server, the client, nor any attacker could use the data exchanged or the data they hold to mimic any other party involved. Zero-knowledge systems are inherently resistant to attacks involving fraudulent servers, clients, and man-in-the-middle attackers because all systems are treated with the same level of distrust.

The first major implementation of a zero-knowledge authentication system was EKE, short for Encrypted key exchange [59]. EKE uses a shared secret between the client and server to negotiate a shared key to be used for encrypted communications. The principles behind EKE became the basis for several variations of EKE [60, 61], including a version that negotiated a key on the basis of a memorable password [62]. Other refinements to EKE have dealt with flaws in the mechanics of the protocol [63, 64]. Figure 1 [1] gives a rough genealogy of EKE and its relation to other zero-knowledge systems.

Following EKE were a new crop of zero-knowledge authentication systems, most notably SRP (Secure Remote Password protocol) [65, 66]. SRP improves on the performance of EKE and adds additional protections from the leak of cryptographic data. While still operating on a simple, memorable password, SRP provided mutual authentication without the use of a third party.

Despite the advancement of zero-knowledge systems, the unique requirements for web-based authentication requires a departure from classical zero-knowledge authentication systems. An effective system for the Web has to compensate for an almost complete exposure of authentication data while at the same time being fast, portable, and utilize only a small portion of the available bandwidth.

B Genealogy of Password Protocol

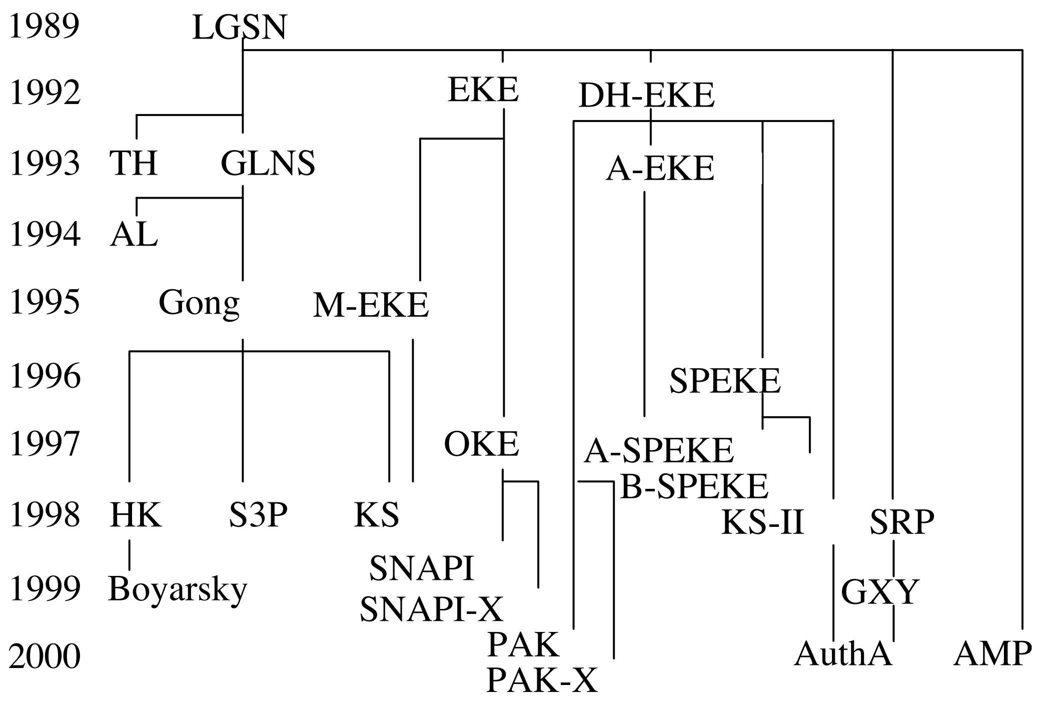

Fig. 1: An overview of EKE and its derivatives [1]

Despite their attractive properties, classical zero-knowledge systems alone do nothing to effectively solve the problem of phishing. Systems like SRP which still use the memorable password as the only requisite information needed for authentication are still be subject to attack by deception. Including phishing protection within a zero-knowledge system requires a change in the type of data used in authentication.

Two-Factor Authentication

One of the secure design principles in set out in the seminal paper by Saltzer & Schroeder [53] was the concept of "separation of privilege." Separation of privilege stated that access should never be granted on the basis of a single condition, this drove the development of the standard username/password authentication system. On a physical terminal the username/password scheme fulfills separation of privilege because both items must be known and entered separately. With remote authentication the username/password model falls apart due to the transmission medium; the username and password become a single sequence of data which can be intercepted and replicated by malicious parties. This compression of the username/password pair into a single authentication condition through remote transmission is the reason why phishing is possible at all.

Remote authentication has been one of the primary motivations for the development of stronger authentication systems. One of the stronger mechanisms developed is referred to as *two-factor authentication*, and uses a combination of authenticating credentials which have no implicit relationship to each other. Two-factor authentication does not solely rely on the traditional username-and-password scheme for identification, but uses another element of a completely different nature which is immune to disclosure through psychological manipulation. Under a two-factor system the disclosure of the a password would be insufficient to authenticate with the server and could not be used to derive the second authentication factor.

Two-factor authentication schemes are resistant to psychological deceptions (like phishing) because they use an authentication factor that is unknown to the user, and so is not subject to disclosure. Username and password can be considered information a user "knows"; in a two-factor identification scheme, the user must also provide something outside the scope of their "knowledge" to prove their identity. This is usually achieved with something a user "has" (e.g., a software token, a key card) or something the user "is", (e.g., biometrics).

While implementations of two-factor authentication commonly use a password memorized by the user as one of the authentication factors, the other factor varies significantly. The first two-factor systems used physical “smart cards” which could be read by specialized hardware and produce one-time keys for granting access [67]. One of the most popular forms of two-factor authentication in use today is biometrics [68, 69, 70], though it has been hampered by the cost and reliability of readers. Current systems focused on remote authentication over a network often use asymmetric cryptography keys as a factor (like Kerberos [71])), while others use software tokens with un-memorizable values [72].

Various two-factor authentication systems for use on the Web have been proposed [73, 74] but have gained little traction. Only banks, the main targets of phishing, have made investment into two-factor authentication system which operate over the Web. The lack of standards and a reliance on additional hardware have limited the effectiveness of these solutions. Two-factor is also sometimes erroneously seen as a magic bullet against attackers, which leads to implementations that defend against phishing but do not counter traditional threats to authentication [75].

Like zero-knowledge, two-factor authentication cannot not prevent the user from disclosing the information they have memorized (i.e., a password), but two-factor does prevent the negative consequences of password disclosure: unauthorized access. In ZeKo, two-factor authentication protects against the consequences of a potentially successful phishing attack while providing all the advantages of a traditional strong authentication mechanism. Currently ZeKo uses a large random value (a *software token*) to implement its second factor (the first being a password), but this could be changed to use any other identifying characteristic (such as biometrics) that is not user memorizable. While the token is essential to authentication, it is never transmitted during the authentication process and its value is unknown to the user. The token's very nature and an unmemorable value makes it immune to disclosure through psychological deception—such as phishing—and any sort of attack on the authentication process.

Setup Procedure

The goal of the setup procedure is to initialize the authentication data at both the client and server for future authentication (illustrated in Figure 2). In particular, the client will acquire two independent authentication elements: a password and a token. The server will obtain a verifier that can be used to establish their own identity and check the client's responses. (Application-specific user account details can also be negotiated during setup, however, this is outside the scope of the protocol.) We assume that prior to setup, there has been some sort of positive identification of the server that the client is performing setup with, and a username has also been agreed upon (otherwise there is a threat of "setup convolution", discussed in Sec. V).

The setup procedure in ZeKo consists of the following steps:

1. The client and the server establish a secure channel (using TLS, for example). This secure channel protects against interception of the authentication data that the client and the server will exchange in the following steps.

2. The server creates a *token* unique to the user by generating a large random number. Immediately after the client receives the token, the server erases the token from its memory, as it is no longer needed by the server. Because we assume the data on the server to be compromised, if the server were to retain the token it could make the user vulnerable to phishing in the future. (It is

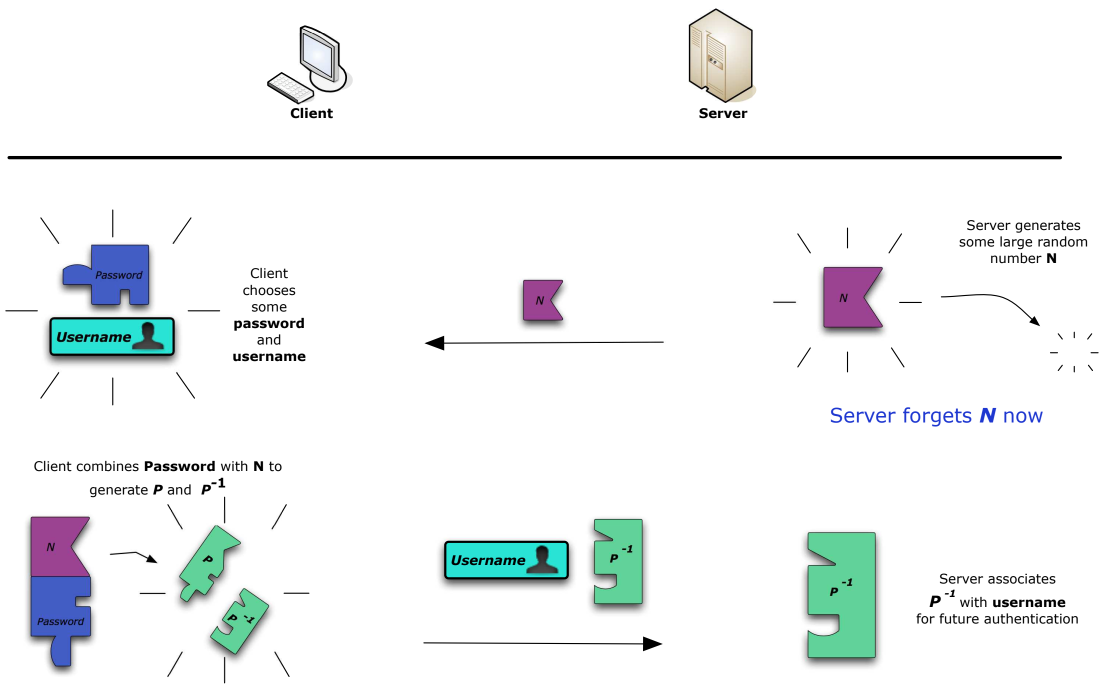
Client
Server
Password
Username
Client chooses some **password** and **username**
N
Server generates some large random number **N**
Server forgets ***N*** now
Client combines **Password** with **N** to generate ***P*** and ***P*⁻¹**
P
P -1
Username
P -1
Server associates ***P*⁻¹** with **username** for future authentication

Fig. 2: ZeKo Setup procedure

still an open question as to whether the client or server should generate the token and how it should be redistributed to other authenticating devices, addressed in more detail in Sec. V.)

3. After the client receives the token from the server, it chooses a password. The password does not necessarily have to be complex or unique in order to be secure, but it should be non-trivial. With the token and the password as the input, the client then uses a strong one-way hash function to calculate a hash value, and uses the hash value as a seed for creating a pair of asymmetric keys, P and P^{-1}. Anything encrypted by P can only be decrypted by P^{-1}, and vise-versa. P will be used by the client for encryption during authentication, and P^{-1} is transmitted to the server through the secure channel. The user can choose to keep both keys on the system for faster future authentication, but for greater security can choose to keep only

the token on the system and recreate the keys with the password as needed for authentication. The server receives P^{-1} as well as the username, and stores the tuple for future authentication.

The setup procedure's security is a result of being conducted through a secure channel. In fact, even in the case of a full disclosure of the contents of the communication, the interceptor does not have enough information to mimic the user. Nonetheless, the full disclosure is not a desirable situation as it can make the client vulnerable to phishing by the same attacker.

The steps for renegotiating the client and server authentication data (like in a lost password/token situation) does not necessarily use this same process. There are several mechanisms for handling the redistribution of the token, such as a specialized procedure using other types of identifiable information but they are beyond the scope of ZeKo's setup procedure. Exploration into the question of how to handle renegotiation of authentication data and distribution of the token to other devices has shown a simple repetition of the setup process to be insufficient, discussed more thoroughly in Sec. V.

Authentication Procedure

The following section outlines the actual process of authentication using ZeKo. For clarity, the authentication process is divided into two phases: (1) authentication of the server to the client (illustrated in Figure 3), and (2) the authentication of the client to the server (illustrated in Figure 4).

In ZeKo the authentication of the server is a necessary condition of the process. Other systems, such as SRP [66], can also provide server authentication, but it is done after client authentication. While continuing to authenticate with a server that has not authenticated itself does not put the client at greater risk, the server's failure to authenticate itself in phase 1 makes phase 2 unnecessary.

All traffic transmitted during the authentication mechanism requires no additional encryption at the transport layer and can be accomplished with UDP messaging. UDP is actually the preferred manner of authentication as it requires less commitment from the user (and server) to an authentication that may fail; this is especially important on mobile devices where every transmission costs battery time. At the end of authentication the conversation can be switched to a different port over TCP at the same time as the transition to the shared symmetric key.

Authenticating Server to Client

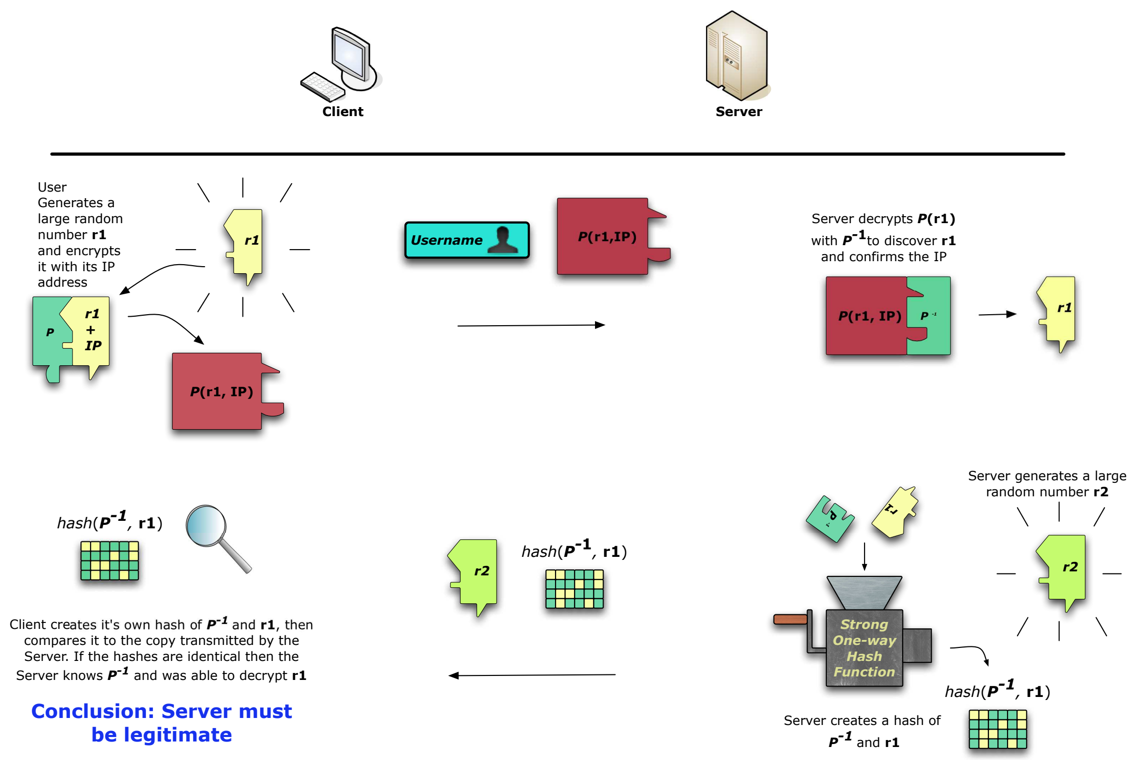
Client
Server
User Generates a large random number r1 and encrypts it with its IP address
r1
P
r1 + IP
P(r1, IP)
Username
P(r1,IP)
Server decrypts P(r1) with P-1 to discover r1 and confirms the IP
P(r1, IP)
P-1
r1
hash(P-1, r1)
Client creates it's own hash of P-1 and r1, then compares it to the copy transmitted by the Server. If the hashes are identical then the Server knows P-1 and was able to decrypt r1
Conclusion: Server must be legitimate
r2
hash(P-1, r1)
Server generates a large random number r2
r2
Strong One-way Hash Function
hash(P-1, r1)
Server creates a hash of P-1 and r1

Fig. 3: ZeKo Phase 1: Authenticating the server to the client.

Just prior to the initiation of authentication, the client recreates P and P^{-1} using the password and token in the same manner as they were originally created in the Setup procedure. While the client could keep a cached copy of P and P^{-1} for later use, in the event the client is compromised the account would be compromised. If only the token is kept on the client it would be impossible to compromise the account by simply stealing the user's stored data since the password is still unknown.

Phase 1 (authenticating the server to the client) consists of two steps:

1. The client creates a large random number unique to this session, r_1 (often called a *nonce*), and encrypts it along with its IP address using P (the asymmetric key generated from the password and token). To make each authentication session unique, r_1 could be composed of a random number and a timestamp. The client then sends the encrypted information (i.e., $P(\text{IP}, r_1)$) along with its username to the server. The IP address is included in order to prevent relay and man-in-the-middle attacks. While a man-in-the-middle may be able to reuse an old authentication message or act as a pass-through for a legitimate user, the IP address in the packet will not match up.

   The server receives the message and decrypts it using P^{-1} discovering r_1 and the IP address. The server checks the IP address to do a preliminary confirmation of the client; an incorrect address being evidence of a relay attack (accommodations being made users operating behind a NAT).

2. The server then uses a strong one-way hash function on r_1 and its own decryption key, P^{-1}. It generates its own unique random number r_2, and transmits it to the client along with the hash it just generated, i.e., $hash(r_1,$

P^{-1}). The client receives the hash and r_2, and checks against a hash of r_1 and P^{-1} it generates itself. An identical hash proves to the client that the server it is communicating with has P^{-1} and was able to successfully decrypt r_1. From this the client can conclude that the server is authentic.

Authenticating Client to Server

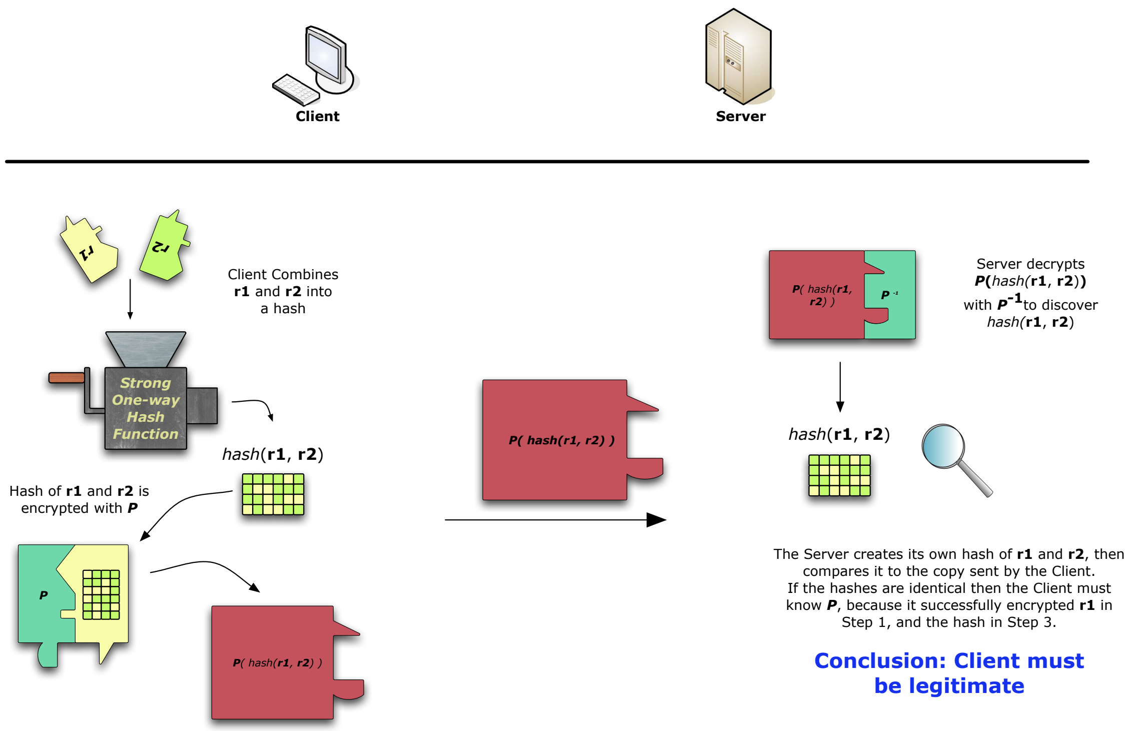
Client
Server
r1
r2
Client Combines r1 and r2 into a hash
Strong One-way Hash Function
hash(r1, r2)
Hash of r1 and r2 is encrypted with P
P
P(hash(r1, r2))
P(hash(r1, r2))
P(hash(r1, r2))
P⁻¹
Server decrypts P(hash(r1, r2)) with P⁻¹ to discover hash(r1, r2)
hash(r1, r2)
The Server creates its own hash of r1 and r2, then compares it to the copy sent by the Client. If the hashes are identical then the Client must know P, because it successfully encrypted r1 in Step 1, and the hash in Step 3.
Conclusion: Client must be legitimate

Fig. 4: ZeKo Phase 2: Authenticating the client to the server.

Phase 2 (authenticating the client to the server) builds on the first phase, and consists of the two steps below:

1. After confirming the server's validity in phase 1, the client hashes together the random numbers generated by both the client and server (r_1 and r_2), and encrypts them with P. It then sends the encrypted hash, i.e., $P(hash(r_1, r_2),$), to the server.

2. The server decrypts and checks the hash from the client with a hash generated by the server consisting of the same elements. If the hash sent is identical to one generated by the server, the client must be the entity who successfully encrypted r_1, and therefore knows P (which was created using the password and token).

In the interest of speed, after the completion of the authentication the server and client will want to use a symmetric key for the remainder of their conversation. This can be done on the basis of elements both the client and server would know and which could not be observed during the authentication process, such as: $hash(P^{-1}(r_1,))$. This is however a sub-optimal solution; in the case that the server data has been compromised it would be possible for a middleman to eavesdrop on the conversation after the transition to a symmetric key. The more secure response would be for the server to send the first packet of data sent to the client encrypted with P^{-1} containing a generated symmetric key for subsequent use.

System Properties

Resiliency against the Theft of Authentication Data

In an adversarial environment like the Internet, any site that a user attempts to access could possibly be a phisher. In order to safely authenticate in this environment, a secure authentication system must allow both parties to definitively prove their identity to each other without disclosing any sensitive information. ZeKo achieves this through the use of zero-knowledge authentication

Though zero-knowledge authentication prevents the theft of the user's secrets during authentication, we still have to assume that users will disclose some of their login information through some other channel, like a deceptive webpage. In order to combat the release of user authentication data, ZeKo merges zero-knowledge authentication with two-factor authentication

ZeKo's use of tokens in collaboration with the password not only protects against the use of trivial passwords, but more importantly, invalidates psychological attacks such as phishing.

Resiliency against Man-in-the-Middle Attacks

The assumption of an adversarial network requires us to be resilient against all forms of man-in-the-middle attack: interception, relay and replay. Zero-knowledge authentication provides us with part of a solution to man-in-the-middle attacks: it protects against the disclosure of sensitive information through interception. However, attacks through direct manipulation of an authentication session (such as relay and replay) are a problem that has to be defeated through careful system design.

ZeKo uses session uniqueness and message source confirmation to stop replay and relay attacks, respectively. Varying certain elements of the authentication process in each session, ZeKo denies the attacker the information needed to impersonate the user by replaying old authentication sessions with the server. The system can also prevent a man-in-the-middle attacker from gaining access by relaying packets from the user by inserting elements into the authentication process (i.e., the IP address in the first message) that confirm the source of the messages.

Resiliency against Cryptanalysis

The goal of any cryptographic attack is to learn enough of a user's authentication data that the attacker can mimic the legitimate user. In any authentication scheme, cryptanalysis could be conducted on the user authentication data, server authentication data, and the authentication traffic. ZeKo is resilient against cryptanalysis by protecting these vulnerable data sets, as well as their combinations:

- **User authentication data:** ZeKo derives its strength from the independence of authentication elements under two-factor authentication: if an attacker steals one element of the user's authentication data, it is impossible to cryptanalyze it to derive the other element. In particular, although phishing may yield the username/password pair to a phisher, the phisher cannot use the password to discover the user's other authentication element and login to a user's account. In short, as long as an attacker does not hold the entire set of a user's authentication data, they cannot then use cryptanalysis to learn how to impersonate the user. (Note that complete knowledge of a user's authentication data would result in a compromised account regardless of the authentication scheme.)
- **Server authentication data:** Just as a user needs a set of authentication data to prove its identity, a server needs its own set of data to confirm the user's identity. For our purposes, the disclosure of secret server data is assumed to be frequent and pervasive with the server always potentially compromised. In this design, stealing the server's authentication data will not make it easier for an attacker to impersonate a valid user and authenticate with the server. The attacker may use the server authentication

data in an attempt to impersonate the server and trick a user into authenticating with them, but that would not provide the attacker with any data that does not still require the solution of a difficult cryptographic problem.

- **Authentication traffic:** When a user authenticates themselves to a server (i.e., a website), the data that the user exchanges with the server provides the richest pool of data for cryptanalysis. The assumption that the network is adversarial means an attacker could have full access to all authentication traffic. Fortunately, zero-knowledge authentication systems, including ZeKo, are specifically designed to conceal secrets, thus are resistant to cryptanalysis on intercepted data [59, 61, 63, 64]. Using state-of-the-art hash functions and cryptographic techniques [76, 77] ZeKo protects all the authentication data that is used in composing messages between a client and a server. ZeKo also ensures that data specific to an authentication session cannot be discovered by attackers through cryptanalysis, preventing the use of replay and selective dictionary attacks.

Resistance to cryptanalysis is build through defense-in-depth. Using just the traffic intercepted during a successful authentication session, an attacker will not be able to derive even the data specific to an authentication session, let alone the authentication data. Even if an attacker can learn some session-specific data, it will be insufficient for the attacker to discover any user or server authentication data. If an attacker discovers the server's authentication data, they will still not be able to obtain the user's authentication data. In the case that an attacker stole one element of the user's authentication data, it will not help the attacker to reveal the other.

While the client and the server successfully authenticate each other through the zero-knowledge authentication process, neither of them disclosed any information to the other—or anyone listening in on the communication—that would compromise the integrity of their private data.

Anti-Phishing Features

In addition to the stated properties of ZeKo discussed previously, there are also several specific features which make it well suited to combatting phishing:

- *Even if a phisher discovers the client's username and password, they cannot login as the client.* The client's username and password are insufficient to derive P or the token, and the token cannot be exposed through phishing since its value is unknown to the user. Because our system is two-factored, the token is entirely unrelated in nature to the password. Possessing either the password or the token tells you nothing about the other. With the client's password but without the client's asymmetric key P, the phisher cannot even perform phase 1 of authentication, where P must be used to encrypt r_1 in order to create a message to the server. The asymmetric key P is also needed in phase 2 to send the client's response to the server.

- *Even if a phisher steals all the authentication data from the server, the phisher cannot login as the client to the legitimate server.* Having the server's authentication data, i.e., P^{-1}, will not help an attacker learn the client's authentication data (i.e., P, the token, or the client's password). Recall that zero-knowledge authentication (Sec. III) does not require the client to expose its secret, so the fake server will not learn the token or P by pretending to be the server and authenticating the client.

- *The attacker cannot use cryptanalysis to learn the client's authentication data, even with all the authentication traffic.* Except the random number r_2 from the server, all ZeKo authentication traffic consists of either strong one-way hash values or messages encrypted with the asymmetric key P. As the strength of the state-of-the-art asymmetric encryption and strong one-way hash functions are well established and still advancing, it is effectively impossible to crypto-analyze the traffic and derive the hash input or asymmetric keys used. Therefore, since the phisher must have both the password and the token to calculate P or have P itself to impersonate the client, the phisher will not be able to login as the client to the legitimate server.

- *Man-in-the-middle attacks are ineffective.* Without knowledge of the server's authentication data, i.e., P^{-1}, the attacker cannot pretend to be the server and authenticate itself to the client. Even with the knowledge of the server's authentication data, as was illustrated above, the attacker will not be able to impersonate the client to login to the legitimate server. The phase 2 message from the client to the server is a hash encrypted with the client's P, and an attacker cannot forge such a message unless they knows both P and r_1.

  As the random number r_1 is unique in each authentication session, it prevents an attacker from locating and replaying an old message from the server in phase 1 (i.e., "hash(r_1, P^{-1}), r_2") in an attempt to impersonate the server. Furthermore, in authenticating the client to the server in phase 2, the message P(hash(r_1, r_2)) also includes a random number r_2 that is unique to the authentication session. This ensures an attacker cannot replay old messages from the client to the server to impersonate the client. With the current setup, the attacker can replay an old message from the client to the server in phase 1 (i.e., $P(\text{IPaddress}, r_1)$); however, the random number r_2 that the server returns in phase 1 is unique to every session, and the attacker will not be able to locate and replay an old message from the client to the server in phase 2 that used the current r_2. thus the attacker will not be able to replay messages from phase 2 (since they will receive a new unique r_2 from the server).

Cryptanalysis based on authentication traffic is also difficult for an attacker. Even when an attacker eavesdrops the entire conversation of every authentication process, they cannot discover anything about the password, the token, P, P^{-1}, or r_1. As said above, except the random number r_2 in the clear, all authentication traffic is protected with either strong one-way hash functions or asymmetric encryption.

CHAPTER IV

EVALUATION

In this section I compare the performance of ZeKo to other authentication mechanisms with similar properties. The evaluation measures the quantifiable properties of each system as well as an in-depth security analysis of the benefits and drawbacks of each system when compared against ZeKo.

Methodology

Evaluated Systems

Given the diversity of authentication systems available for evaluation, it was necessary to select systems for comparison with the most similar security properties to ZeKo. Because the proposed system is a combination of both two-factor and zero-knowledge authentication is is reasonable to select a representative from each of those groups, combined with a username/password system to act as a baseline; each system provides some of the features of ZeKo through the use of a different authentication mechanism. Using a side-by-side comparison we can better understand how effective zero-knowledge and two-factor authentication systems are by themselves and analyze the contribution the ZeKo has to make.

The zero-knowledge representative is SRP [66], the most current and widely used decedent of the EKE lineage discussed in Section III. SRP provides mutual authentication and leak-free key negotiation to its users while also being resilient against the loss of certain parts of its authentication data. I represent two-factor authentication with WikID [78], which is a publicly available, open source product with a well-defined authentication procedure. Operating on top of a secure connection, WikID uses a combination of password and unique asymmetric key shared between the client and server. For a baseline system we also use a simple username/password authentication system over TLS [79].

The Examined Metrics

An effective web authentication mechanism requires not just robust security but a specific set of performance attributes. Given that web access is being included into a diverse range of platforms and devices, few assumptions can be made as to the resources available to perform the authentication. The conservative approach to the development of a new authentication system must therefore seek to minimize the load it puts on both the client device and the network being used. All security conditions being equal, the authentication solution which consumes the least resources can be considered optimal.

The most important concern for devices which have limited connectivity to the authentication server is the minimization of network usage. The network load can be characterized independently of the underlying network conditions by looking at the total amount of data sent and received during the authentication process as well as the number of round trips (RT) the authentication mechanism makes to complete the process. Used in conjunction with the analysis of computational load, network load can provide a realistic estimate of the total time required for authentication given differing network conditions.

To measure the load on the device computationally I used the execution time required for (1) the client to contact and complete the authentication procedure, (2) the time needed by the server to process the request from the client and grant or deny access, and (3) the total time needed to complete the transaction — from the initiation by the client to the final step in the authentication process. All the timing experiments were conducted over the test-bed machine's internal loopback address with both the server and client running on the same machine, minimizing any effect of network conditions on latency.

The other load conditions that act upon a device during authentication are the memory costs of performing the authentication steps as well as the storage cost required to keep the data needed to perform the authentication. While memory calculations fluctuate through the life of the program, the average memory used is sampled throughout the life of the execution cycle and the maximal amount of memory used give a good estimate of the peak and average cost. Memory analysis is restricted to the client program, which is assumed to have greater restrictions on resources then the server. Storage cost is a calculation of everything needed by the authentication mechanism that cannot be actively remembered by the user (i.e., username and password), such as asymmetric keys and identifying tokens.

Security Comparisons

When compared to the baseline system, username/password over TLS, ZeKo provides a significantly greater level of security against phishing and deception attacks. The use of TLS protects the submission of the username/password against active eavesdroppers, but does not inherently identify the receiving server. The certificate systems in PKI which is supposed to fulfill the role of positively identifying the server are completely separate from the authentication process, with TLS only protecting against man-in-the-middle attacks. The failure of this system to stop phishing is discussed in greater detail in Section II

Representing a two factored approach, WiKID attempts to combat phishing through the use of asymmetric keys. The WikID system is designed as a separate authentication server which issues access tickets to other servers in the form of *passcodes*, shared secret byte streams. The WiKID authentication server uses the combination of a *Device ID* and a PIN submitted by the user to establish the user's identity, this process is protected by the server's asymmetric key and immune from phishing. Once the server has confirmed the Device ID and PIN to be valid it responds to the user with a time-sensitive passcode which will be recognized by the server the user actually wants to access. The fundamental flaw in this system is that the usage of the passcode is not standardized, if this passcode is used in a similar fashion to a username/password system then the system is no more secure against phishing then username/password over TLS. While there is a guarantee about the authenticity of the WiKID authentication server, there is no guarantee about the authenticity of server on which the passcode will be used, or how it will be used. Because of this lack of identification of the server receiving the passcode and how it is used, WiKID cannot be considered completely immune to phishing.

The most similar system to ZeKo in mechanism is SRP. SRP is a zero-knowledge protocol based which performs a symmetric key negotiation with the user in a similar manner to the Diffie-Hellman [57] system, but is dependent upon a pre-existing relationship between the two parties. In SRP, a client requests a stored *salt* (a large nonce) from the server which it combines with a secret password to generate a shared value with the server. Through the exchange of specially formulated messages both the client and the server solve an equation with provides mutual verification of authenticity and generates a shared symmetric key. Like ZeKo, SRP is resistant to man-in-the-middle attacks and the negative effects of server data disclosure, but is not two-factored and is vulnerable to phishing. Since SRP authentication requires only the standard username/password pair along with the salt provided by the server to authenticate, the acquisition of the username/password through phishing would be sufficient to access the account as the server distributes the salt to all that ask for it.

Among the surveyed systems ZeKo provides all the security benefits of the other systems while providing greater protections against phishing then any of the other systems. The performance of ZeKo aside, there are sufficient arguments for the adoption of ZeKo solely on the security gains it provides.

A summary of the security properties of each system is presented in Table 1

	Man-in-the-middle Attacks	Cryptographic Strength	Phishing Protection
TLS	High	Medium	Low
WikID	High	High	Medium
SRP	High	High	Low
ZeKo	High	High	High

Tab. 1: Security Properties Summary

Experimental Conditions

In order to examine only the authentication mechanism used by the evaluated systems, each was implemented in Java based on the posted specification without any additional functionality. The authentication procedures for SRP and WiKID were described in detail in their respective documents [66, 78] and the process of username/password authentication using one-way hashes is well known. The establishment and maintenance of the secure connection for systems which used TLS (WiKID and username/password) is handled by the java library for secure connections: `javax.net.ssl`. The initial setup procedures needed by the systems to distribute authentication data for future authentication is handled by a separate program and not included in this analysis.

The choice to implement the systems in Java was motivated in part by the capabilities of Sun's Hotspot JVM [80] which provides real-time monitoring of memory usage during execution. The calculation of memory usage is based on samples of the Java heap size collected 10 ms apart throughout the life of client program. For our purposes the maximal sample is considered to be the maximal memory requirement. Timing is calculated using the system timer functionality and measures the time in milliseconds from the retrieval of authentication data off the hard disk to the receipt/transmittal of the final authentication message.

All of the systems evaluated used an asymmetric key in one form or another and so the size of that key was standardized to be 1024 bits, an accepted standard for key lengths. The RSA asymmetric key algorithm [58] (as implemented by the Java Library) was used for generation of the keys as well as encryption and decryption. All of the systems also used one-way hashes during authentication and this procedure was standardized to use the MD5 [81] (as implemented by the Java Library).

Three of the systems (ZeKo, SRP and username/password) used passwords and usernames as factors in the authentication, the size of both the passwords and usernames was standardized to 40-bits to reflect the findings of a large-scale survey of website passwords [82]. WiKID had several authentication elements which had no direct parallel in the other systems, but which filled the roles of username and passwords, these were also standardized to 40 bits. WikID was also alone in the generation and use of a symmetric key during the authentication procedure, this key was generated using the AES [83] standard with the commonly accepted size of 128 bits. Both ZeKo and SRP used nonces during the authentication procedure, these were both standardized to be 24 bytes.

All of the authentication programs analyzed were run on a Apple Power Mac with dual 1.25 GHz G4 processors and 512 MB of RAM. The code was developed and tested using Java and HotSpot JVM Ver. 1.5 on top of Apple's OS X 10.4.11. Whenever possible the network and memory usage of the programs was minimized and shared mechanisms, such as encryption/decryption, were handled by the same procedures.

Results

Results are summarized in Table 2. The presented results are the the arithmetic mean (rounded to the nearest whole number) of 10 complete authentication experiments of each system.

ZeKo scored consistently better then WiKID and username/password over TLS in terms of network usage and memory costs. ZeKo also had a lower associated storage cost then every system except username/password. SRP scored better then ZeKo in most categories due to the fact that the implementation performs the asymmetric encryption and decryption mathematically, and not through the Java libraries. The results suggest that ZeKo provides superior performance to all the surveyed systems except SRP, which does not provide the

Tab. 2: ZeKo Evaluation Results

	Networking (Bytes)				Timing (ms)			Memory (KBytes)		Storage (Bytes)	
	C $\rightarrow$ S	S $\rightarrow$ C	Total	RT	Client	Server	Total	Max	Avg	Client	Server
TLS	416	239	655	3	1444	234	1445	2397	2322	0	21
WikID	400	581	981	3.5	1573	304	1528	2398	2202	152	160
SRP	89	146	235	2	130	82	132	615	573	128	197
ZeKo	261	63	324	1.5	1538	757	1544	1444	957	128	133

same level of protection against phishing as ZeKo.

The greater analysis of the results are presented below:

Networking

SRP had the lowest overall networking usage for bytes sent and received, though made slightly more RTs then ZeKo due to an extra message needed to authenticate the server to the client (the last step in SRP). ZeKo's higher byte count when compared to SRP mostly due to the increased use of encryption in ZeKo's first and third messages. For cryptographic reasons, the size of the encrypted data is always a multiple of the size of the key. When the data to be encrypted is not an exact multiple of the key size, the data is "padded" to reach the size of the next multiple. This padding present in the usage of encrypted data added to the size of the transmitted messages in ZeKo that was not present in SRP.

ZeKo scored better then WiKID and the username/password system primarily because of the overhead required for the initiation of the TLS connection both used. The setup and tear-down cost of TLS on both systems was 644 Bytes and 2.5 RTs. This secure connection cost was larger then the cost of the authentication procedure itself of the systems that used it; authentication alone cost 11 Bytes + 0.5 RT for username/password and 337 Bytes + 1 RT for WiKID. Because WiKID also used encryption for its messages it also experienced some padding of the data transmitted.

The username/password scheme transmitted a total of 11 Bytes for its authentication which is greater then the expected 10 Bytes (40 bit username, 40 bit password); the extra byte acts as the length field that would be needed to separate the username and password if their lengths were not known prior to authentication.

Timing

ZeKo's worst performance was in the amount of time needed to complete the authentication. Much of this is due to increased amount of encryption and decryption that ZeKo performs using asymmetric keys when compared to the other systems. In total ZeKo performs 2 encryptions, 2 decryptions and 4 hashing operations; only WiKID does a similar number of encryption/decryptions and consequently is closest to ZeKo in terms of time needed to authenticate speed.

As expected SRP did the best due to the low-level implementation of its key generation. The username/password system was the second fastest due to the fact it only needed to perform a single hashing function to complete its authentication.

Memory Usage

Again SRP outperformed all other systems due to the speed of its low-level implementation, requiring only the storage and calculation of large numbers. ZeKo used less average and peak memory then the other two systems primarily due to the memory overhead associated with the TLS connection.

Both WiKID and username/password show similar max memory sizes with suggests that the establishment of the secure connection was the most memory intensive procedure used by each of the programs. The higher average usage of the username/password system is most likely due to the shorter execution span of the program and the lack of delay in response from the server. The average memory usage while WiKID is waiting for server authentication drops, skewing the overall average lower then username/password which experiences no server delay.

Storage Costs

Predictably the storage cost for username/password was the least, as the username and password are stored by the user and the server only needs to maintain a hash of the password and a username. ZeKo required less storage then the other systems, though came very close to SRP. SRP requires slightly more space on the server to accommodate the “salt” which it distributes to users in the first step of authentication. WiKID also used less space on the server then SRP but required more space on the client because of the use of several long ID tags related to the server and device being authenticated.

CHAPTER V

CONCLUSIONS

To conclude this work I will explore the applications for ZeKo authentication outside of the Web and associated areas of research related to this topic followed by a final review of ZeKo's contributions.

ZeKo Applications

ZeKo lends itself not only to combating phishing on the Web, but also to any application that needs strong, leak-proof authentication over an open transmission medium (such as the Internet or a wireless network). The protocol can modify the size of its keys and nonces to maximize security, performance, or bandwidth conservation (depending on the context). For example, a lightweight version of ZeKo could be used on mobile devices, or larger keys could be used in enterprise networks to provide more protection against data disclosure.

Mobile devices hold some of the greatest promise for ZeKo. Many mobile devices desire to take advantage of open wireless hotspots to conduct texting and voice communication that does not utilize cell service. The cost in battery power of wireless transmissions and the security concerns of connecting to untrusted networks are two main restrictions to implementing this technology. Because ZeKo uses rather small UDP packets for the initial server authentication it requires a minimal investment in battery power by the mobile device to test the network's connectivity. The system also assumes the mobile device is connecting to a hostile network, so there is no security risk in attempting to initiate authentication over the network.

Associated Areas of Research

During the investigation of ZeKo I encountered several implementation issues which mirror larger questions related to phishing and authentication. These issues provide a roadmap for exploring other elements of network security that have a direct impact on anti-phishing authentication.

Authentication Setup

One of the dangers surrounding the implementation of an authentication system involves *setup convolution*. Setup convolution is a situation where an attacker manipulates the initial exchange of authentication data during account setup in such a way that both the user and attacker have access to the account on the legitimate server, leaving the user with no clue that the account has been compromised. The most obvious way of performing setup convolution is to lure a potential user to an attacker's site using an e-mail that encourages joining, or through DNS manipulation. Once at the attackers site, the user undergoes a setup process identical to the one that they would encounter at the legitimate site. All responses given by the user during setup could be relayed to the legitimate site where an identical setup is taking place. There are several solutions to prevent setup convolution as described above, but setup convolution is only a sample of a more significant problem.

An exploration of setup convolution brings up fundamental questions about account setup in an untrustworthy environment. The most conventional answer to this problem is to use certificates authorized by some trusted third party to verify a site's authenticity; but this solution only works if comprehensively implemented and enforced. Browsers allow users the option of accepting untrusted certificates or sending data to a website over unencrypted channels, a functionality which is needed for some legitimate purposes but which still poses a threat to the account. The question becomes: how to ask users for information in a way that cannot be exploited? Is there a way of asking for information which also confirms the asker? What are the general requirements for sending data in such a way that only an intended party has access when there is no obvious way to identify the intended party? These question all fall in line with the investigation about how to conduct authentication without trust.

The Out-of-System Mechanism (OSM)

Apart from the problem of initially distributing the authentication data is the problem of redistribution. While the elements used in authentication can be distributed to the client securely during startup, we also have to consider situations in which they need to be redistributed. A user may forget passwords or lose authentication tokens, leaving them unable to authenticate with the server. The authentication elements need a simple, yet secure, mechanism through which they can be redistributed. Redistribution of authentication elements after setup would have to be performed outside of the authentication system itself since a secure mechanism could not be performed without those elements. This question of how to distribute authentication data after setup is actually a question of how to restore a legitimate user access to their account without authentication. I have termed the method of distributing authentication data outside of the setup procedure as the *out-of-system mechanism*, or OSM.

The simplest OSM for our system seems to be to distribute the token via e-mail in parallel to the setup procedure. The prevalence of web-accessible e-mail makes for a highly accessible way to distribute to other systems. This OSM raises problems because it is susceptible to interception, and access to the e-mail account may too be susceptible to phishing. Another solution may be a specialized token re-negotiation process based on an independent confirmation of identity. Naturally, the re-negotiation would have to be as secure and resistant to phishing as the rest of the system.

OSM appears at first glance to be just an implementation issue, but it is actually part of a broader discussion about the nature of secure relationships. The question about how best to redistribute the elements of ZeKo is actually a question about how the attributes of the authentication elements effect the nature of secure relationships. An investigation of the interplay of authentication element attributes and their effect on the security of the system provides insight into what essential requirements are needed for a secure relationship.

Current systems have to rely on home-grown analysis tools to argue for their security and cannot examine the entirety of the threats. If there were a rigorous definition of the elements and attributes necessary for a secure relationship, then protocols could be proven secure in a formal way before even being implemented. Exploration of the OSM problem combined with a systematic security analysis of the protocol could provide a good basis for exploring the fundamental nature of secure systems.

ZeKo's Contributions

ZeKo provides an advancement over traditional zero-knowledge systems such as SRP and EKE through the use of a mutual authentication scheme that requires less round trips and total messages transmitted. ZeKo also contributes to two-factor authentication through the integration of the second factor into the messaging process; while most two factor schemes use the second factor as an additional password, ZeKo uses only the combination of the elements together for authentication making them harder to intercept or disclose.

Overall, ZeKo provides significant improvement over the username/password system used in web authentication today and has anti-phishing advantages not provided by other strong authentication systems. In addition to the unique security benefits, ZeKo also requires less overhead then comparable systems. The combination of two-factor and zero-knowledge authentication provided here is a significant step forward in addressing the root causes of phishing.

REFERENCES

[1] T. Kwon, "Authentication and key agreement via memorable password," Tech. Rep. 2000/26, 2000. [Online]. Available: citeseer.ist.psu.edu/article/kwon00authentication.html

[2] E. J. Dean Turner, Stephen Entwisle, "Symantec internet security threat report: Trends for january–june 07," Symantec, Tech. Rep., September 2007.

[3] Anti-Phishing Working Group, "Phishing activity trends report for the month of June, 2007," 2007. [Online]. Available: http://www.antiphishing.org/reports/apwg_report_june_2007.pdf

[4] M. Jakobsson and J. Ratkiewicz, "Designing ethical phishing experiments: A study of (ROT13) rOnl query features," in *Proceedings of the international conference on World Wide Web*, 2006, pp. 513–522.

[5] National Consumers Leauge, "A call for action: Report on the National Consumers Leauge anti-phishing retreat," March 2006.

[6] D. Florencio and C. Herley, "A large-scale study of web password habits," in *Proceedings of the international conference on World Wide Web*. New York, NY, USA: ACM Press, pp. 657–666.

[7] "Gartner study finds significant increase in e-mail phishing attacks," Press release, May 2004. [Online]. Available: http://www.gartner.com/press_releases/asset_71087_11.html

[8] R. Johannes, "2006 identity fraud survey report," Javelin Strategy & Research, Tech. Rep., 2006.

[9] "Gartner survey shows phishing attacks escalated in 2007; more than $3 billion lost to these attacks," Press release, Dec 2007. [Online]. Available: http://www.gartner.com/it/page.jsp?id=565125

[10] M. Smiles. (2007) New message from oregon community credit union - oregon community credit union phishing scams. [Online]. Available: http://www.millersmiles.co.uk/report/5967

[11] (2008) University of oregon becomes a "phishing" target: Bogus emails emulate duckweb login. [Online]. Available: http://it.uoregon.edu/news/phishing_target.shtml

[12] "Identity theft resource center facts and statistics," 2007, visited October 11, 2007. [Online]. Available: http://www.idtheftcenter.org/artman2/publish/m_facts/Facts_and_Statistics.shtml

[13] "An inquiry into the nature and causes of the wealth of internet miscreants," in *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security.* New York, NY, USA: ACM, pp. 375–388.

[14] A. Elledge, "Phishing: An analysis of a growing problem," SANS Institute, Tech. Rep., 2007.

[15] F. Paget, "Identity theft," McAfee Avert Labs, Tech. Rep., January 2007.

[16] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," in *Proceedings of Conference on Human Factors in Computing Systems*, 2006, pp. 581–590.

[17] J. S. Downs, M. B. Holbrook, and L. F. Cranor, "Decision strategies and susceptibility to phishing," in *Proceedings of the Symposium on Usable Privacy and Security*, 2006, pp. 79–90.

[18] W. Goerigk, "On trojan horses in compiler implementations," in *Proc. des Workshops Sicherheit und Zuverlassigkeit softwarebasierter Systeme*, 1999.

[19] (2003) How interactive logon works. [Online]. Available: http://technet2.microsoft.com/windowsserver/en/library/779885d9-e5e9-4f27-9c14-5bbe77b056ba1033.mspx?mfr=true

[20] M. Laureano, C. Maziero, and E. Jamhour, "Intrusion detection in virtual machine environments," in *EUROMICRO '04: Proceedings of the 30th EUROMICRO Conference.* Washington, DC, USA: IEEE Computer Society, 2004, pp. 520–525.

[21] S. Stamm, Z. Ramzan, and M. Jakobsson, "Drive-by pharming," Indiana University, Tech. Rep., 2006. [Online]. Available: http://www.symantec.com/avcenter/reference/DrivebyPharming.pdf.

[22] Mozilla Foundation, "Firefox phishing protection," visited October 10, 2007. [Online]. Available: http://www.mozilla.com/en-US/firefox/phishing-protection/

[23] Microsoft Corp., "Anti-phishing technologies overview," visited October 10, 2007. [Online]. Available: http://www.microsoft.com/mscorp/safety/technologies/antiphishing/overview.mspx

[24] E. Kirda and C. Kruegel, "Protecting users against phishing attacks with AntiPhish," in *Proceedings of the International Computer Software and Applications Conference*, 2005, pp. 517–524.

[25] K.-P. Yee and K. Sitaker, "Passpet: Convenient password management and phishing protection," in *Proceedings of the Symposium on Usable Privacy and Security*, 2006, pp. 32–43.

[26] M. Wu, R. C. Miller, and S. L. Garfinkel, "Do security toolbars actually prevent phishing attacks?" in *Proceedings of Conference on Human Factors in Computing Systems*, 2006, pp. 601–610.

[27] J. A. Halderman, B. Waters, and E. W. Felten, "A convenient method for securely managing passwords," in *WWW '05: Proceedings of the 14th international conference on World Wide Web*. New York, NY, USA: ACM, 2005, pp. 471–479.

[28] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell, "Stronger password authentication using browser extensions," in *SSYM'05: Proceedings of the 14th conference on USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2005, pp. 2–2.

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

[30] CallingID Ltd., "CallingID Toolbar," visited October 10, 2007. [Online]. Available: http://www.callingid.com/DesktopSolutions/CallingIDToolbar.aspx

[31] Netcraft, "Netcraft anti-phishing toolbar," visited October 10, 2007. [Online]. Available: http://toolbar.netcraft.com/

[32] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails," in *Proceedings of the international conference on World Wide Web*, 2007, pp. 649–656.

[33] B. Adida, S. Hohenberger, and R. L. Rivest, "Fighting phishing attacks: A lightweight trust architecture for detecting spoofed emails," in *Proceedings of DIMACS Workshop on Theft in E-Commerce: Content, Identity, and Service*, Apr. 2005.

[34] Y. Zhang, J. I. Hong, and L. F. Cranor, “Cantina: A content-based approach to detecting phishing web sites,” in *Proceedings of the international conference on World Wide Web*, pp. 639–648.

[35] A. Y. Fu, W. Liu, and X. Deng, “Detecting phishing web pages with visual similarity assessment based on earth mover’s distance (EMD),” *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 4, pp. 301–311, 2006.

[36] A. Rosiello, E. Kirda, C. Kruegel, and F. Ferrandi, “A layout-similarity-based approach for detecting phishing pages,” in *International Conference on Security and Privacy in Communication Networks*, 2007.

[37] L. Cranor, S. Egelman, J. Hong, and Y. Zhang., “Phinding phish: An evaluation of anti-phishing toolbars.” Carnegie Mellon University, Tech. Rep. CMU-CyLab-06-018, Nov 2006.

[38] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, “A comparison of machine learning techniques for phishing detection,” in *Proceedings of the eCrime Researchers Summit*, 2007, pp. 60–69.

[39] N. Agarwal, S. Renfro, and A. Bejar, “Yahoo!’s Sign-in Seal and current anti-phishing solutions,” in *Proceedings of Web 2.0 Security & Privacy Workshop*, May.

[40] Bank of America, “How Bank of America SiteKey works for online banking security,” visited October 10, 2007. [Online]. Available: http://www.bankofamerica.com/privacy/sitekey/

[41] The Vanguard Group, “Learn how Vanguard protects your accounts,” visited October 10, 2007. [Online]. Available: https://personal.vanguard.com/VGApp/hnw/help/SecurityVGProtectsAcctsContent.jsp

[42] Tricerion, “Tricerion SMA product overview,” visited October 10, 2007. [Online]. Available: http://www.tricerion.com/products/productOverview.php

[43] R. Dhamija and J. D. Tygar, “The battle against phishing: Dynamic security skins,” in *Proceedings of the Symposium on Usable Privacy and Security*, 2005, pp. 77–88.

[44] S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer, “The emperor’s new security indicators,” in *SP ’07: Proceedings of the 2007 IEEE Symposium on Security and Privacy.* Washington, DC, USA: IEEE Computer Society, 2007, pp. 51–65.

[45] Earthlink, Inc., "Earthlink tools for Firefox browser," visited October 10, 2007. [Online]. Available: http://www.earthlink.net/software/nmfree/firefox/faq/

[46] eBay, Inc, "eBay: Buyer's tools: Toolbar," visited October 10, 2007. [Online]. Available: http://pages.ebay.com/ebay_toolbar/

[47] Google, Inc., "Google safe browsing for Firefox," visited October 10, 2007. [Online]. Available: http://www.google.com/tools/firefox/safebrowsing/

[48] C. Ludl, S. McAllister, E. Kirda, and C. Kruegel, "On the Effectiveness of Techniques to Detect Phishing Sites," in *Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, 2007.

[49] J. S. Downs, M. Holbrook, and L. F. Cranor, "Behavioral response to phishing risk," in *Proceedings of the eCrime Researchers Summit*, 2007, pp. 37–44.

[50] P. Kumaraguru, Y. Rhee, S. Sheng, S. Hasan, A. Acquisti, L. F. Cranor, and J. Hong, "Getting users to pay attention to anti-phishing education: evaluation of retention and transfer," in *Proceedings of the eCrime Researchers Summit*, 2007, pp. 70–81.

[51] Binational Working Group on Cross-Border Mass Marketing Fraud, "Report on phishing: A report to the Minister of Public Safety and Emergency Preparedness Canada and the Attorney General of the United States," October 2006.

[52] B. J. Fogg, J. Marshall, O. Laraki, A. Osipovich, C. Varma, N. Fang, J. Paul, A. Rangnekar, J. Shon, P. Swani, and M. Treinen, "What makes web sites credible?: a report on a large quantitative study," in *Proceedings of Conference on Human Factors in Computing Systems*, 2001, pp. 61–68.

[53] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278 – 1308, 1975.

[54] S. Hartman, "Requirements for web authentication resistant to phishing," Nov. 2007, IETF Internet-Draft. [Online]. Available: http://www.ietf.org/internet-drafts/draft-hartman-webauth-phishing-06.txt

[55] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, 1989.

[56] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems," *Journal of the ACM*, vol. 38, no. 3, pp. 690–728, 1991.

[57] E. Rescorla, "Diffie-Hellman Key Agreement Method," RFC 2631, IETF, 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2631.txt

[58] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[59] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secureagainst dictionary attacks," in *SP '92: Proceedings of the 1992 IEEE Symposium on Security and Privacy.* Washington, DC, USA: IEEE Computer Society, 1992, p. 72.

[60] ——, "Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise," in *ACM Conference on Computer and Communications Security*, pp. 244–250.

[61] M. Steiner, G. Tsudik, and M. Waidner, "Refinement and extension of encrypted key exchange," *Operating Systems Review*, vol. 29, no. 3, pp. 22–30, 1995.

[62] D. P. Jablon, "Strong password-only authenticated key exchange," *Computer Communication Review*, vol. 26, no. 5, pp. 5–26.

[63] C.-L. Lin, H.-M. Sun, and T. Hwang, "Three-party encrypted key exchange: attacks and a solution," *Operating Systems Review*, vol. 34, no. 4, pp. 12–20, 2000.

[64] E. Bresson, O. Chevassut, and D. Pointcheval, "Security proofs for an efficient password-based key exchange," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security.* New York, NY, USA: ACM, 2003, pp. 241–250.

[65] T. Wu, "The secure remote password protocol," in *Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium*, pp. 97–111.

[66] ——, "Srp-6: Improvements and refinements to the secure remote password protocol," 2002, submission to the IEEE P1363 Working Group.

[67] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *CRYPTO '89: Proceedings on Advances in cryptology.* New York, NY, USA: Springer-Verlag New York, Inc., 1989, pp. 239–252.

[68] J. Armington, P. Ho, P. Koznek, and R. Martinez, "Biometric authentication in infrastructure security," in *InfraSec '02: Proceedings of the International Conference on Infrastructure Security.* London, UK: Springer-Verlag, 2002, pp. 1–18.

[69] T. Connie, A. Teoh, M. Goh, and D. Ngo, "Palmhashing: a novel approach for dual-factor authentication," *Pattern Analysis & Applications*, vol. 7, no. 3, pp. 255–268, 2004.

[70] A. T. B. Jina, D. N. C. Linga, and A. Gohb, "Biohashing: two factor authentication featuring fingerprint data and tokenised random number," *Pattern Recognition*, vol. 37, no. 11, pp. 2245–2255, 2004.

[71] J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)," RFC 1510, IETF, 1993, obsoleted by RFC 4120. [Online]. Available: http://www.ietf.org/rfc/rfc1510.txt

[72] D. N. Hoover and B. N. Kausik, "Software smart cards via cryptographic camouflage," in *IEEE Symposium on Security and Privacy*, 1999, pp. 208–215.

[73] N. Zhang, J. Chin, A. Rector, C. Goble, and Y. Li, "Towards an authentication middleware to support ubiquitous web access," in *COMPSAC '04: Proceedings of the 28th Annual International Computer Software and Applications Conference - Workshops and Fast Abstracts - (COMPSAC'04).* Washington, DC, USA: IEEE Computer Society, 2004, pp. 36–38.

[74] T. Verschuren, "Smart access: strong authentication on the web," in *TNC'98: Proceedings of the TERENA networking conference '98 on Towards networking and services in the year 2001.* Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V., 1998, pp. 1511–1519.

[75] B. Schneier, "Two-factor authentication: too little, too late," *Commun. ACM*, vol. 48, no. 4, p. 136, 2005.

[76] S. Kent and K. Seo, "Security architecture for the Internet Protocol," RFC 4301, IETF, 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4301.txt

[77] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," in *Proceedings of the Symposium on Theory of Computing*, 1989, pp. 33–43.

[78] M. Mitchell and E. Shoemaker, "Wikid technology white paper, ver 1.4," Tech. Rep., December 2005. [Online]. Available: http://superb-east.dl.sourceforge.net/sourceforge/wikid-twofactor/WiKID_White_Paper_v1.4.pdf

[79] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," RFC 2246, IETF, 1999, updated by RFC 3546. [Online]. Available: http://www.ietf.org/rfc/rfc2246.txt

[80] (2006) Java se hotspot at a glance. Sun Microsystems. [Online]. Available: http://java.sun.com/javase/technologies/hotspot/

[81] R. Rivest, "The MD5 message-digest algorithm," RFC 1321 (Informational), IETF, 1992. [Online]. Available: http://www.ietf.org/rfc/rfc1321.txt

[82] D. Florencio and C. Herley, "A large-scale study of web password habits," in *Proceedings of the international conference on World Wide Web.* New York, NY, USA: ACM Press, 2007, pp. 657–666.

[83] NIST, "Federal information processing standards publication 197: Announcing the advanced encryption standard," 2001.