# An Anti-Phishing Password Authentication Protocol

Pramote Kuacharoen
Department of Computer Science, Graduate School of Applied Statistics
National Institute of Development Administration
118 SeriThai Rd., Bangkapi, Bangkok 10240, Thailand
(Email: pramote@as.nida.ac.th)

*(Received Aug. 31, 2016; revised and accepted Jan. 15 & Feb. 20, 2017)*

## Abstract

Password authentication is commonly used to authenticate the user in web-based services such as internet banking due to its simplicity and convenience. Many users have multiple accounts and use the same password. The password is usually sent to the server over an HTTPS connection. However, this common practice makes the system vulnerable. An attacker can set up a phishing site masquerading as the genuine site and attempts to steal the user's credentials. If the user's credentials are successfully stolen, all accounts are compromised. Moreover, since passwords are common, a break-in to a system that is not well protected might cause a cascaded break-in. This paper describes an authentication protocol which enables the user to securely use the same password for multiple servers, and protects against phishing attacks. The protocol also allows multiple authentication sessions simultaneously while preventing replay attacks. Furthermore, the protocol is also resilient against denial-of-service attacks since no state is maintained on the server during the authentication process.

*Keywords: Anti-phishing; Authentication; Mutual Authentication; Password*

## 1 Introduction

Phishing is a technique that employs both social engineering and technical subterfuge to steal personal identifiable information and financial account credentials. The criminal creates a replica of an existing web page to deceive the victims [3]. Usually, the criminal sends emails which resemble emails from legitimate entities to potential victims. Unaware of criminal activities, the victims click the link in the email to visit the website where they are asked to provide personal information such as username, password, and credit card number. The criminal records this information and uses it to impersonate the victims or to commit financial fraud [15, 16].

Although the phishing site appears to be similar to the legitimate site, the Uniform Resource Locator (URL) is different, usually suspicious. The phishing site is short lived so that it cannot be effectively blacklisted. An experimental phishing attack was performed at Indiana University targeting students aged 18 to 24 years old [9]. The acquaintance data are harvested from social network websites. The experiment spoofed an email message between two friends. Experiments showed that 72% of students entered their secure university credentials into the spoofed site whose domain name was clearly distinct from Indiana University.

When the user moves the pointer to hover over a hyperlink, the URL is usually shown on the status bar. A user with this knowledge makes an attempt to verify the destination URL using this method as a safeguard against phishing. However, a status bar can be easily spoofed. The criminal can use a simple *onclick* event to change the destination URL.

Many web browsers have anti-phishing features built in. However, some users fail to notice the warning, do not understand the warning, or ignore the warning [5]. In order to provide the anti-phishing features, the web browsers must maintain the list of the phishing sites. As aforementioned, phishing sites cannot be effectively blacklisted and the user is not protected until the phishing site is included on the list.

Several large financial institutions, including Bank of America and The Vanguard Group, attempt to combat against phishing attacks by implementing a technique called SiteKey which is the product of RSA Data Security. SiteKey uses the following challenge-response technique:

1) The customer identifies himself by submitting the username. If the username is valid, the site continues to the next step. Otherwise, the site displays an error message indicating that the username is not correct.

2) The site authenticates itself to the customer by displaying an image and a phrase that the user has previously chosen. If the user does not recognize them, the user should assume that the site is a phishing

site and should not proceed. If the user recognizes the displayed information, the user may consider that the site is authentic.

3) The user authenticates oneself by supplying the password. If the password is correct, the user is authenticated.

In practice, SiteKey is ineffective [19, 24]. People do not notice or do not care when the SiteKey is missing. Moreover, SiteKey technique has a security design flaw. The criminal can learn whether or not the username exists. The rationale of SiteKey is that the phishing site does not have the customer's SiteKey. However, the phishing site can obtain the correct SiteKey from the genuine site, and then displays it to the user.

The Anti-Phishing Working Group (APWG) reported that phishing attack numbers declined 20 percent from late 2012 to early 2013. This was due to a precipitous drop in virtual server phishing attacks, where the criminal seizes control of a web server that hosts many unique domains and then creates phishing pages for those domains [7]. According to APWG, trends indicate phishing levels returning to the levels seen prior to the record-setting highs of 2015. Therefore, these criminal activities are still prevalent and an effective anti-phishing attack technique is needed.

The purpose of this research is to design and implement an authentication protocol which is secure and protects the user against phishing attacks. The following requirements are the design goals of the anti-phishing password authentication protocol.

- The protocol must protect users against phishing attacks.
- The protocol must allow users to safely use the same password across many websites.
- The protocol must achieve user authentication without reviewing the password to the server at any point.
- The protocol must be secure against known attacks.

This paper consists of five sections. Section 1 introduces the motivation of the paper. Section 2 describes background information and related work in the area of phishing and password authentication. Section 3 presents the design of the anti-phishing password authentication protocol. Section 4 provides the security analysis of the protocol. Finally, Section 5 concludes the paper.

## 2 Background and Related Work

This section provides background and related work which includes phishing, password authentication, cryptographic challenge-response authentication, and existing anti-phishing password-based protocols.

### 2.1 Phishing

Phishing is the attempt to obtain sensitive information such as usernames, passwords, and financial information by masquerading as a trustworthy entity in electronic communication [14]. Phishers use social engineering schemes using spoofed emails purporting to be from legitimate businesses and agencies. The schemes are designed to lead victims to counterfeit websites and deceive the victims into divulging sensitive information.

APWG publishes quarterly phishing attack trends reports. Figure 1 shows the phishing trends. The total number of unique phishing reports received has sharply risen from year 2012 to year 2015. Phishing has been increasingly threatening individuals.

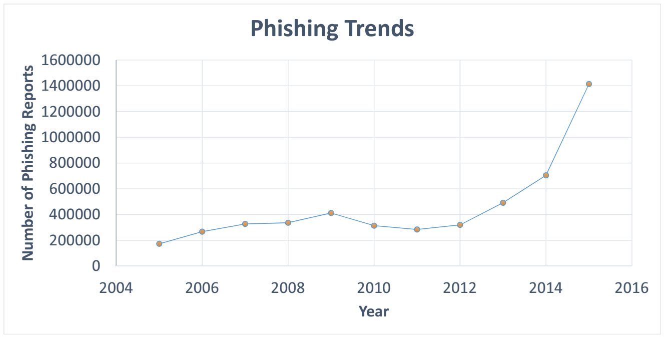
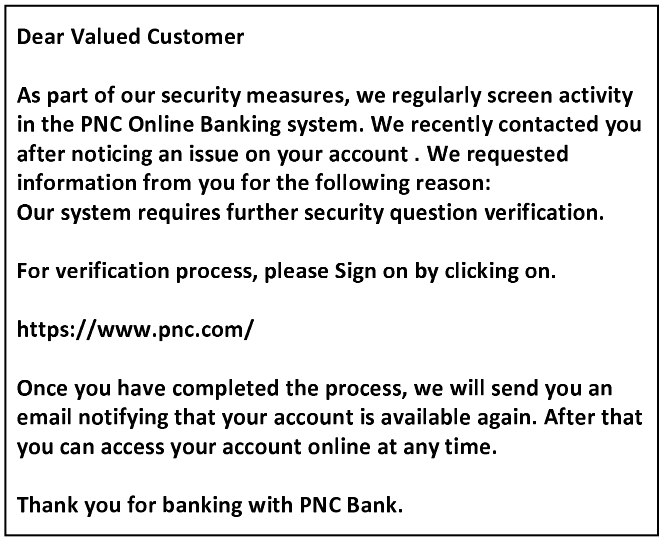


Figure 1: Phishing trends

Phishers usually send emails impersonating trusted entities luring victims to visit phishing sites. Phishing targets the user who has no knowledge about social engineering attacks or internet security [8]. Figure 2 shows an example phishing email which pretends to be from PNC Bank asking its customer to sign in by clicking on the link. The link displays the URL of PNC Bank. However, when the victim clicks on the link, the phishing site is shown. An unsuspecting customer who has an account with PNC Bank would sign in as instructed in the email. By doing so, the customer unknowingly gives the phisher their account credentials.

For an HTML page, the displayed link and the actual link can be different. When user moves a pointer over the link, the status bar shows the actual URL. This may give some confidence to the user who is familiar with browsing the Web. However, a status bar can be programmed to display whatever the phisher desires. A user with some technical knowledge is likely to be a victim for a status bar spoofing. When the user clicks the link, the actual link shows on the browser's address bar. The phisher tricks the victim by using a site name similar to the real site by misspelling the name. For example, the phisher may use letter 'a' instead of letter 'o', '1' instead of letter 'l', or '0' instead of letter 'o'. When the user glances at the address bar, the user assumes the website is legitimate. The phisher may also employ a poorly written redirection program from the real website to the phishing site. The



**Dear Valued Customer**

**As part of our security measures, we regularly screen activity in the PNC Online Banking system. We recently contacted you after noticing an issue on your account . We requested information from you for the following reason:**
**Our system requires further security question verification.**

**For verification process, please Sign on by clicking on.**

**https://www.pnc.com/**

**Once you have completed the process, we will send you an email notifying that your account is available again. After that you can access your account online at any time.**

**Thank you for banking with PNC Bank.**

Figure 2: An example of phishing email

address bar will display the real website address briefly and then it will display the phishing site address. This will make the victim think that displayed website is redirected from the trusted site and trust the displayed site.

Online security warnings have historically failed because users do not understand or believe them. Since phishing is prevalent, new online security warnings have been redesigned. It is difficult to automatically detect phishing with accuracy. Therefore, anti-phishing tools use warnings to alert the users to potential phishing sites, rather than blocking them. There are two types of warnings, namely, passive warning and active warning. A passive warning indicates a potential danger by changing colors, providing textual information, or by other means without interrupting the user's task. However, research has shown that passive warnings are failing users because users often fail to notice them or do not trust them. An active warning, on the other hand, forces users to notice the warning by interrupting them. Many popular web browsers include active phishing warnings since research has shown that passive warnings are often ignored.

Despite growing efforts to educate users and to create better detection tools, users are still susceptible to phishing attacks. Phishing can deceive users because the users are willing to trust websites that appear to be designed well and look familiar.

Many research papers have proposed methods of detecting phishing sites based on URLs of web pages [1, 4, 10, 23]. After confirming that the website is probably a phishing site, a security warning is issued to the user. However, using phishing site detection methods does not guarantee that the algorithms are always accurate.

## 2.2 Password Authentication

Password authentication is the simplest form of an authentication model. The user presents the username and the password to the authenticating entity [13, 20]. The password authentication is commonly used in authenticating the user over the Internet. The server needs to store the user's password in order to authenticate the user [17]. The password must be protected. Storing the password in clear text is inadvisable. This is because a compromised user database file reveals all passwords. The attacker may be able to obtain the password database through an SQL injection attack [2, 12]. Instead, the hash value of the password should be saved. A weakness of using the hash value is that two users with the same password have an identical hash value. Furthermore, the attacker can use a dictionary attack against the entire user database. The best way to protect a password is to employ salted password hashing. The server randomly generates a number called salt and calculates the hash of the salt and the password. Therefore, two users with the same password have different salted passwords. The server stores the salt and the salted password along with other information. Upon logging in, the user supplies the username and the password. The server computes the hash value of the salt and the received password, and then compares the resulting value to the stored hash value. If the two values are identical, the user is authenticated.

Sending a password in clear text is vulnerable to eavesdropping. Using an SSL connection helps protect the conversation during transit. However, it does not prevent phishing attacks.

## 2.3 Cryptographic Challenge-Response Authentication

In this type of authentication, the user and the system share some secret [21]. For two-way authentication, both the user and the system must convince each other that they know the shared secret without transmitting the secret in the clear text over the communication channel. To accomplish this, the server encrypts the randomly generated number or nonce and sends it to the user as a challenge. The user must return a corresponding response which is calculated from decrypting the challenge and encrypting the value derived from the decrypted value. This proves that the user has the ability to decrypt the challenge. Therefore, the user knows the shared secret.

A drawback of the challenge-response authentication is that it can be defeated by man-in-the-middle attacks. For example, the user visits a phishing site and submits his username. The phishing site forwards the user's identity to the genuine site impersonating the user. The server sends a challenge in order to authenticate the user. The attacker presents the challenge to the user to obtain the correct response which is subsequently sent to the server.

## 2.4 Existing Anti-phishing Password-based Protocols

Rose et al. present a method to improve password security and to defend against password phishing [18]. The server stores the hash of the user's password and the domain name. When the user enters the username and the password is prefixed with two escape characters, the browser extension applies a Pseudo Random Function (PRF) to a combination of the password and the domain name. The username and the hash value are sent to the server. The domain name is automatically obtained. If the user enters the credentials on the phishing page, the phisher cannot obtain the clear text password. Moreover, the hash value is different from the one stored on the actual server since the phishing site has a different domain name. Using domain name as salt has a drawback. The attacker may compromise a server under the same domain and may set up a phishing page. The correct salted password can be captured. Since password and domain name remains unchanged, the salted password is the same, making it susceptible to replay attacks.

In [6], a protocol that allows a client to securely use a single password across multiple servers and prevents phishing attacks is proposed. The client can be authenticated without revealing the password to the server at any point. The protocol employs a one-time ticket technique. The client sets the next authentication ticket. The ticket consists of the hash of the random number, the password, and the server name. The client identifies himself by sending the identity. The server challenges with the previously stored random number. Subsequently, the client computes the ticket using the received number. The client also randomly generates a number and uses it to create the next authentication ticket. The client responds with the current ticket, the next challenge random number, and the hash value of the next authentication ticket. Although the clear text password remains unchanged, the ticket changes each time the client is authenticated. This is equivalent to changing the password at the server every time the user signs in, which makes the protocol susceptible to message modification attacks. Consider the scenario where an attacker intercepts the response from the client. The attacker then can create a ticket using his password and replaces the hash value with the one generated from his ticket. The server has no way to verify the authenticity of the hash value. Hence, the attacker can log in.

## 3 Design of the Protocol

This section describes the design of the anti-phishing password authentication protocol. The design objectives and the notions are explained. Then, this section discusses the password storage and the authentication that are designed to meet the objectives.

### 3.1 Design Objectives

The primary objective of this paper is to design an authentication protocol which is secure and protects the user against phishing attacks and other known attacks. The following requirements are the design goals of the anti-phishing password authentication protocol.

1) The protocol must protect users against phishing attacks.

2) The protocol must allow users to safely use the same password across many websites.

3) The protocol must achieve user authentication without revealing the password to the server at any point.

4) The protocol must be secure against known attacks such as password database attacks, server spoofing attacks, and denied of service attacks.

5) The protocol must allow multiple authentication sessions.

6) The protocol must be mutual authentication.

### 3.2 Notations

Table 1 shows notations which are used throughout this paper.

Table 1: Notations and description

| Notations | Descriptions |
|---:|---|
| $C$ | Client |
| $S$ | Server |
| $AD_S$ | Server's address, i.e., IP address |
| $AD_C$ | Client's address, i.e., IP address |
| $N_1, N_2, N_3$ | Nonce |
| $HMAC(K, M)$ | Keyed-hash message authentication code function, where $K$ is the secret key and M is the message |
| $Times$ | Session valid time which consists of the start time and the expiration time |
| $SAC$ | Session authentication code |
| $\|\|$ | Concatenation operator |
| $\oplus$ | XOR operator |

### 3.3 Password Storage

It is crucial that the user's credentials are protected even though the user database has been compromised. The attacker should not gain knowledge from it. Therefore, the password should not be stored or sent as clear text. Traditionally, for each user, the server stores the username, the salt, and the hash value of the salt and the user's password. This protects the user's credentials and defends against dictionary attacks and pre-computed rainbow table attacks. However, both username and password are sent to the server to be authenticated. For a valid username, the server calculates the hash value of the salt and the received password and compares the resulting value with the one stored on the server. Since the actual password is transmitted to the server, the attacker can obtain this information from a compromised server and can use the user's credentials to gain access to other servers. Therefore, the user's password should not be transmitted as clear text.

If the password is hashed at the client machine and sent to the server, the client-side hash logically becomes the user's password. Therefore, it is equivalent to storing passwords in clear text. If the attacker obtained the hash value, the attacker can use it to authenticate to the server. Hence, the server must store a value which is derived from the received value.

To achieve the design objectives, the authenticator must be derived from the user's password and must be server specific. Table 2 shows the user database. Each row consists of username, salt, and masked secret. The salt is credential specific. In other words, each user is randomly assigned a salt. This prevents a dictionary attack on the entire database. The attacker must pick an individual to attack. The masked secret is calculated by XORing the user's secret and the mask. The user's secret is derived from the username, the password, and the Fully Qualified Domain Name (FQDN), i.e., Hash(username || password || FQDN). The mask is the hash value of the server's secret and the user's salt, HMAC($K_S$ || salt). The server's secret is not stored on disk. It is inputted when the server starts. Therefore, a compromised database does not reveal the server's secret or the user's secret.

HMAC is a message authentication code based on a cryptographic hash function [11]. The length of the authentication code is fixed. Only the parties, who have the knowledge of the secret key, can produce a valid message authentication code. The advantage of using HMAC is that the cryptographic hash function generally executes more quickly in software than symmetric and asymmetric ciphers [22].

Table 2: User database

| Username | Salt | Masked Secret |
|---|---|---|
| $u_1$ | $salt_1$ | $K_{c1} \oplus HMAC(K_s, salt_1)$ |
| $u_2$ | $salt_2$ | $K_{c2} \oplus HMAC(K_s, salt_2)$ |
| $u_3$ | $salt_3$ | $K_{c3} \oplus HMAC(K_s, salt_3)$ |
| ... | ... | ... |

## 3.4 The Protocol

In an insecure network environment, any client can connect to a server. The obvious risk is user impersonation. An attacker can pretend to be another user and obtain unauthorized access. To counter this threat, the server must be able to authenticate the user requesting the service.

Figure 3 summarizes the basic authentication dialog. The following is the brief description of the protocol.

1) The client identifies itself to the server by sending the user's ID, the address of the server, and a random value $N_1$.

2) The server replies back with received information, another random value $N_2$, and the start time and the expiration time of the authentication session. Moreover, the server includes the server's authenticator and the Session Authentication Code (SAC).

3) The client responds with the user's ID, server address, $N_2$, another random value $N_3$, $Times$, the client's authenticator, and the session authentication code.

**1. C$\rightarrow$ S: $ID_C||AD_S||N_1$**
**2. S$\rightarrow$ C: $ID_C||AD_S||N_1||N_2||Times||Authenticator_S$ || SAC**
**3. C$\rightarrow$ S: $ID_C||AD_S||N_2||N_3|Times||Authenticator_C||SAC$**
**$Authenticator_S = HMAC(K_C, ID_C||AD_S||N_1||N_2||Times)$**
**$Authenticator_C = HMAC(K_C, ID_C||AD_S||N_2||N_3||Times)$**
**$SAC = HMAC(K_S, ID_C||AD_C||N_2||Times)$**

Figure 3: Summary of the authentication exchanges

When the user connects to a server to use a service, the client software $C$ in the user's computer requests the username and password, and then sends a message to the server $S$ that includes the user's ID, the server's address, and the random nonce $N_1$. The server first verifies if the requested server's address belongs to the server, and checks its database to see if the user exists. If the user is in the database, the server obtains the salt and the masked secret of the user. The mask is generated from hashing the server's secret and the user's salt. The resulting value is then XORed with masked secret to obtain the user's secret. Next, the server randomly generates another nonce $N_2$ and sets the start time and the expiration time for the authentication session.

The server constructs the response message which includes user's ID, the server's address, the received nonce $N_1$, the generated nonce $N_2$, and the times. The server authentication which is used to authenticate itself to the client is computed using the HMAC algorithm with the response message and the user's secret. The session authentication code for the session is also generated using the HMAC algorithm with the server's secret and the message which includes the user's ID, the client address, the nonce $N_2$, and the times. The server sends the response message, the server authentication, and the session authentication code. After replying, the server can discard the calculated values.

When the client receives the response, the client verifies that the response is corresponding to the request by checking if the response message contains the requested information. The server authenticator is then verified. A valid server authenticator proves that the server knows the user's secret. Now that the client has the challenge information, authenticating itself can be done next. The client creates a response message which consists of the user's ID, the server address, the received nonce $N_2$, the nonce $N_3$, and the times. The client also creates the client authentication by using the HMAC algorithm with the response message and the user's secret key. Subsequently, the client sends the response message, the client authenticator, and the session authentication code to the server.

Upon receiving the response, the server checks to ensure that the response is intended for the server and has not expired. If the user's ID is in the database, the server derives the user's secret from the masked secret, the user's salt, and the server's secret as previously described. Afterward, the server verifies the client authenticator. Successful verification implies that the user knows the password which is a crucial component in creating the client secret. The server then validates the session authentication code. A valid SAC is the SAC which has not expired and has not been used. The server saves the SAC which has been used to the user's SAC list. Future sessions will be checked against this list to ensure that the SACs are used only once. To perform the validation task efficiently, expired SACs are removed from the SAC list.

Table 3 summarizes the justification for each of the elements in the protocol.

## 4 Security Analysis

This section analyzes the security of the proposed protocol which includes the security of passwords, security of the server's secret key, and security of the communication protocol.

### 4.1 Security of Passwords

The obvious approach to password attack is to guess the password. The two most common methods of guessing passwords are brute-force attacks and dictionary attacks. These two types of attacks can be performed online or offline as described in the following section.

#### 4.1.1 Online Brute-Force and Dictionary Attacks

In the brute-force approach, the attacker tries all possible passwords. On average, an attacker will have to try half of all possible combinations before finding the correct password. To defend against such an attack, the password length policy must be enforced. The password must be at least eight characters long. A longer password is a better password. Moreover, for online password guessing, the system should be configured to slow the attack by delaying between successive login attempts and limiting the number of unsuccessful attempts before disabling the account for a period of time or indefinitely until the account is reset.

The attacker impersonates the user and attempts to log in on a server by trying all passwords in an exhaustive list called a dictionary. An online dictionary attack feeds a server with thousands of username and password combinations. To protect against dictionary attacks, the user must use a strong password which can be enforced by access policy. Guidelines that are designed to make passwords less easily discovered by intelligent guessing and cracking tools include using complex passwords with an appropriate length. A complex password is a password which uses several types of keyboard characters. As a result, complex passwords are unlikely to be in the attacker's dictionary. The number of unsuccessful login attempts should also be limited as aforementioned.

#### 4.1.2 Offline Brute-Force and Dictionary Attacks

The attacker obtains the user database and attempts to perform offline brute-force attacks or dictionary attacks. Each entry in the user database consists of the username, the salt, and the masked secret. The masked secret is calculated from the user's secret and the mask. However, the mask is user specific and depends on the server's secret. Therefore, this makes it impossible to use lookup tables and rainbow tables to crack the password.

### 4.2 Security of the Server's Secret Key

The server's secret key is used to create the user mask and the session authentication code using the HMAC algorithm. The user mask is not stored in the user database. It is XORed with the user's secret. If the attacker obtains the user database, the mask is not readily available. However, the attacker can register an account. Since the attacker can compute his own secret key, the mask can be obtained. Breaking the server's secret key would only compromise all users' secrets on a specific server. However, if the attacker breaks the server's secret key without having the user database, the attacker will not be able to impersonate other users. The security of the server's secret key depends entirely on the security of the HMAC algorithm.

Attacks on HMAC can be grouped in two categories, namely; brute-force attacks and cryptanalysis. The level of effort for brute-force attack on the HMAC algorithm has a similar level to that for symmetric encryption algorithms. Cryptanalysis attacks on the HMAC algorithm. The security of the HMAC depends in some way on the cryptographic strength of the underlying hash function. HMAC is considered secure. Therefore, it is computationally infeasible for the attacker to derive the server's secret key.

### 4.3 Security of the Communication Protocol

The authentication exchanges are done over the HTTPS protocol. However, the proposed authentication protocol

Table 3: Rationale for the elements of the protocol

| | |
|---:|---|
| **Message 1** | Client requests an authentication session |
| $ID_C$ | Tells the server's identity of the user from this client |
| $AD_S$ | The perceived server's address by the client, i.e., the IP address |
| $N_1$ | A random value to be repeated in message 2 to assure that the response is fresh and has not been replayed by the attacker |
| **Message 2** | Server returns a session and authenticates itself to the client |
| $ID_C$ | Indicates the rightful owner of the session |
| $AD_S$ | The server's address |
| $N_1$ | Nonce from message 1 |
| $N_2$ | A random value to ensure that the response is fresh. It is also used to generate the ticket authentication code |
| $Times$ | Provides time sensitive authentication |
| $Authenticator_S$ | Proves that the server knows the client's secret and the information has not been modified |
| $SAC$ | The session authentication code to be repeated in message 3 to ensure that the session is authentic and is used only once within the time limit |
| **Message 3** | Client authenticates itself to the server |
| $ID_C$ | Indicates the rightful owner of the ticket |
| $AD_S$ | The client's perceived server address which allows the server to have many addresses |
| $N_2$ | Nonce from message 2 to prevent a replay |
| $N_3$ | A random value to ensure that response is fresh. It is also used to generate the session authentication code |
| $Times$ | The value from message 2 to provide a time period of the session |
| $Authenicator_C$ | Proves that the user knows the password and the information has not been modified |
| $SAC$ | The value from message 2 is used to verify that the message is authentic |

is also analyzed when the authentication exchanges are done over an insecure channel.

#### 4.3.1 Eavesdropping Attacks

Since the authentication exchanges are performed over the HTTPS protocol, the eavesdropper cannot obtain the authentication messages. Hence, the attacker cannot learn any information. Without a secure connection, the authentication exchanges may be eavesdropped and the attacker is able to obtain messages. The attacker cannot learn the password because it is not sent to the server. However, non-secret values including the ID of the user, server's address, nonce, and times are revealed. Learning these values does not make the protocol vulnerable. The remaining values are the server's authenticator, the client's authentication, and the session authentication code. These values are generated using HMAC. As previously discussed, HMAC is secure. Therefore, the protocol is secure against eavesdropping attacks.

#### 4.3.2 Message Replay Attacks

The SSL/TLS communication is protected against replay attacks using MAC which is computed using the secret and the sequence number. Therefore, the replayed message is detected as a duplicate. Replaying the entire session is not possible since the master secret is generated using the pre-master secret, the client and the server's random data. Even without SSL, the protocol prevents the replay attacks. When the user has been authenticated, an attacker may be able intercept the message in Step 3 and establishes another session. Since unexpired SACs are saved, the replayed message will contain a used SAC. Therefore, the attacker's session will not be authenticated. Hence, the protocol protects against replay attacks.

#### 4.3.3 Message Modification Attacks

For this type of attack, an attacker attempts to modify a message transmitted between the client and the server to discover the client's password or to gain unauthorized access. Modifying an SSL data stream will cause an error in the packet. The attacker will not gain knowledge of the user's password or unauthorized access. For an insecure channel, the message exchanges in Step 2 and Step 3 are protected by the authenticator and the session authentication code. Without the knowledge of the password and the server's key, the verification will fail.

#### 4.3.4 Denial of Service Attacks

This paper limits the scope of the denial of service attacks to the level of authentication, not the underneath layers. The proposed authentication is stateless which means that the server does not need to remember any challenge values. There is no extra resource reserved for the user. The server can securely and correctly verify the user in Step 3. The server challenge information is implicitly calculated into the server authentication code.

#### 4.3.5 Phishing Attacks

In phishing attacks, the attackers make an attempt to obtain sensitive information such as user credentials and credit card details. Phishing is typically carried out by social engineering techniques such as email spoofing and instant messaging to deceive users. The victims receive fraudulent messages which appear to come from a trustworthy entity. The message usually directs the victim to an authentic-looking website which lures the victim to enter sensitive information. The proposed method protects the user against phishing attacks. Since the attacker does not have knowledge of the user's credentials, it cannot create a valid server authenticator. The address of the attacker is different from the address of the server. Verification will not be successful.

#### 4.3.6 Man in the Middle Attacks

In this type of attack, the traffic between the client and the server goes through the attacker. The attacker is capable of capturing the traffic, modifying it, and replaying the modified traffic. This may be in the form of active phishing where the attacker entices the victim to enter confidential information on the impersonated website and the attacker actively modifies the information and supplies it to the server. In Step 1, the server's address will be the attacker's address. This is because the server's address is automatically obtained. The attacker will modify it to the real server's address. In Step 2, all values in clear text can be altered. However, the server authenticator and session authentication code cannot be modified without the client's secret key and the server's secret key, respectively. Therefore, when the user verifies the server authenticator, the verification process results in failure.

## 5 Conclusions

Password-based authentication is still widely used. However, it may be vulnerable to phishing attacks. The proposed protocol attempts to address this issue by implementing mutual authentication which both client and server must prove that they know the shared secret. A two-factor authentication which includes the knowledge factor and location factor is also used. The proposed protocol also applies a challenge-response authentication in which both server nonce and client nonce are used. This ensures that previous authentications cannot be reused in replay attacks. Moreover, the protocol utilizes timestamps to ensure exact timeliness. Finally, the server can be implemented in a stateless manner during the authentication.

In order to protect the user against phishing attacks, when the user initiates the login process, the user's secret is dynamically derived from the username, the password, and FQDN. If the user is on the phishing site, the user's secret is generated incorrectly. As a result, the verification process would safely fail. The attacker is not able to obtain the user's credentials or perform a transaction on behalf of the user. Therefore, the proposed anti-phishing password authentication protocol automatically protects users from attackers who try to obtain the user's password or make transactions against the user's interest as illustrated in the security analysis.

## Acknowledgments

This study was supported by the National Institute of Development Administration (NIDA), Thailand. The author gratefully acknowledges the anonymous reviewers for their valuable comments.

## References

[1] A. A. Ahmed and N. A. Abdullah, "Real time detection of phishing websites," in *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON'16)*, pp. 1–6, Oct. 2016.

[2] M. Štampar, "Inferential sql injection attacks," *International Journal of Network Security*, vol. 18, no. 2, pp. 316–325, 2016.

[3] J. Chen and C. Guo, "Online detection and prevention of phishing attacks," in *Proceedings of The First International Conference on Communications and Networking in China*, pp. 1–7, Oct. 2006.

[4] A. Y. Daeef, R. B. Ahmad, Y. Yacob, and N. Y. Phing, "Wide scope and fast websites phishing detection using urls lexical features," in *2016 3rd International Conference on Electronic Design (ICED'16)*, pp. 410–415, Aug. 2016.

[5] S. Egelman, L. F. Cranor, and J. Hong, "You've been warned: An empirical study of the effectiveness of web browser phishing warnings," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'08)*, pp. 1065–1074, New York, NY, USA, 2008.

[6] M. G. Gouda, A. X. Liu, L. M. Leung, and M. A. Alam, "Spp: An anti-phishing single password protocol," *Computer Networks*, vol. 51, pp. 3715–3726, Sept. 2007.

[7] Anti-Phishing Working Group, *Phishing Activity Trends Report*, Dec. 2016. (http://www.antiphishing.org/resources/apwg-reports/)

[8] S. Gupta, A. Singhal, and A. Kapoor, "A literature survey on social engineering attacks: Phishing attack," in *Proceedings of the International Conference on Computing, Communication and Automation (ICCCA'16)*, pp. 537–540, Apr. 2016.

[9] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer, "Social phishing," *Communication of ACM*, vol. 50, pp. 94–100, Oct. 2007.

[10] A. K. Jain and B. B. Gupta, "Comparative analysis of features based machine learning approaches for phishing detection," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom'16)*, pp. 2125–2130, Mar. 2016.

[11] H. Krawczyk, M. Bellare, and R. Canetti, *Hmac: Keyed-hashing for Message Authentication*, Technical Report RFC 2104, Internet Engineering Task Force (IETF), Feb. 1997.

[12] P. Kuacharoen, "A practical customer privacy protection on shared servers," in *Proceedings of the IEEE International Conference on Information Theory and Information Security*, pp. 525–529, Dec. 2010.

[13] I. Liao, C. Lee, and M. Hwang, "A password authentication scheme over insecure networks," *Journal of Computer and System Sciences*, vol. 72, no. 4, pp. 727–740, 2006.

[14] A. S. Martino and X. Perramon, "Phishing secrets: History, effects, and countermeasures," *International Journal of Network Security*, vol. 11, no. 3, pp. 163–171, 2010.

[15] A. A. Orunsolu, A. S. Sodiya, A. T. Akinwale, B. I. Olajuwon, M. A. Alaran, O. O. Bamgboye, and O. A. Afolabi, "An Empirical Evaluation of Security tips in Phishing Prevention: A Case Study of Nigerian Banks," *International Journal of Electronics and Information Engineering*, vol. 6, no. 1, pp. 25–39, 2017.

[16] A. A. Orunsolu, A. S. Sodiya, A. T. Akinwale, B. I. Olajuwon, "An Anti-Phishing kit Scheme for Secure Web Transactions," *International Journal of Electronics and Information Engineering*, vol. 6, no. 2, pp. 72–86, 2017.

[17] E. O. Osei, J. B. Hayfron-Acquah, "Cloud computing login authentication redesign," *International Journal of Electronics and Information Engineering*, vol. 1, no. 1, pp. 1–8, 2014.

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

[19] S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer, "The emperor's new security indicators," in *Proceedings of the IEEE Symposium on Security and Privacy (SP'07)*, pp. 51–65, May 2007.

[20] R. Shirey, *Internet Security Glossary*, Technical Report RFC 2828, Internet Engineering Task Force (IETF), May 2000.

[21] W. Stallings, *Cryptography and Network Security: Principles and Practice (7ed)*, Hoboken, NJ: Pearson, 2016.

[22] P. Subpratatsavee and P. Kuacharoen, *Transaction Authentication Using HMAC-Based One-Time Password and QR Code*, pp. 93–98, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[23] Y. Xue, Y. Li, Y. Yao, X. Zhao, J. Liu, and R. Zhang, "Phishing sites detection based on url correlation," in *2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS'16)*, pp. 244–248, Aug. 2016.

[24] J. Youll, *Fraud Vulnerabilities in Sitekey Security at Bank of America*, July 18, 2006. (`http://cr-labs.com/publications/SiteKey-20060718.pdf`)

## Biography

**Pramote Kuacharoen** received his B.S. and M.E. degrees in computer and systems engineering from Rensselaer Polytechnic Institute (RPI) in 1995 and 1996, respectively. He also received his M.S. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology in 2001 and 2004, respectively. He joined the Department of Computer Science at National Institute of Development Administration in 2004. His research interests include computer and network security, information security, computer networks, embedded systems, and mobile applications design and development.