Chapter 21
PAKE on the Web

Xunhua Wang
James Madison University, USA

Hua Lin
University of Virginia, USA

ABSTRACT
Unlike existing password authentication mechanisms on the web that use passwords for client-side authentication only, password-authenticated key exchange (PAKE) protocols provide mutual authentication. In this article, we present an architecture to integrate existing PAKE protocols to the web. Our integration design consists of the client-side part and the server-side part. First, we implement the PAKE client-side functionality with a web browser plug-in, which provides a secure implementation base. The plug-in has a log-in window that can be customized by a user when the plug-in is installed. By checking the user-specific information in a log-in window, an ordinary user can easily detect a fake log-in window created by mobile code. The server-side integration comprises a web interface and a PAKE server. After a successful PAKE mutual authentication, the PAKE plug-in receives a one-time ticket and passes it to the web browser. The web browser authenticates itself by presenting this ticket over HTTPS to the web server. The plug-in then fades away and subsequent web browsing remains the same as usual, requiring no extra user education. Our integration design supports centralized log-ins for web applications from different web sites, making it appropriate for digital identity management. A prototype is developed to validate our design. Since PAKE protocols use passwords for mutual authentication, we believe that the deployment of this design will significantly mitigate the risk of phishing attacks.

INTRODUCTION
Phishing attacks have become more widespread these days. A study by the Anti-Phising Working Group found that the password stealing malicious code URLs increased from 9529 in June of 2008 to 31173 (more than tripled) in December of 2008 (Anti-phishing Working Group, 2009). Another study showed that phishing cost the U.S. $3.2 billion in 2007 (Hodgin, 2007). Among the vic-
tims of phishing attacks, financial institutions are hit hardest. Because of the rising online identity theft caused by phishing attacks and regulators’ concerns, financial institutions are looking for better on-line security tools against phishing attacks (Richmond, 2005).

In a typical phishing attack, a user first receives an email informing him to update his online account (such as an online banking account or an online utility bill account) using an embedded URL. The email also warns that failing to conform would result in account lock-out or deletion. Struck by panic, the user may immediately click on the given link, which would lead the user’s web browser to a web site that bears the right logo and a familiar appearance. The user then hastens to type in his account name and password. But actually that web site is run by an attacker, who now has the information to perform fraudulent transactions in that user’s name. Phishing attacks harvest secret information for client-side authentication by impersonating a web server.

Most of today’s web applications use three methods to authenticate a client, namely, basic authentication, digest authentication, and form-based authentication. (There is a fourth client authentication method on the web, the certificate-based client authentication. In this method, a client is assigned a private key, which is used in the Secure Socket Layer (SSL) protocol for client authentication. This method is not common for home users as it requires a user to manage a random private key and ordinary users lack the expertise to handle a private key correctly.) All these three methods are password-based authentications, in which a client remembers a password and the server stores a related password verification data (PVD). In basic authentication, a client’s password is encoded with the (public) Base64 (Josefsson, 2003) method and sent to the server for verification. In digest authentication, the client’s password is not sent to the server; rather, the server sends a random challenge to the client, who then responds with a value calculated from its password and the challenge value. The server compares the received response value with a value calculated from the challenge and the corresponding PVD. In form-based authentication, a client’s password is encapsulated in a HTML form and then transmitted in HTTP to the server.

Should the web server be impersonated, in basic authentication and form-based authentication, the phony web server will receive the client’s password. If digest authentication is used, the phony web server will not get the password directly but having a (challenge, response) pair will allow it to mount off-line dictionary attacks: a value calculated from a guessed password and the challenge is compared against the received response; this process is repeated until a match is found, indicating that the current guessed password is the correct one.

To provide server-side authentication, the common practice is to run all these three password authentications on the top of HTTPS, which is HTTP over Secure Socket Layer (SSL). In HTTPS, each server has a domain name and a public/private key pair. Its public key is certified by some certification authorities (CA), whose public keys have already been shipped with popular web browsers such as Microsoft Internet Explorer (IE) and Mozilla. The resulting digital certificate contains the web server’s public key, its domain name, and a digital signature by the CA covering both the server’s public key and the domain name. After receiving a digital certificate from a server, the client uses the corresponding CA’s public key to verify it. In this way, the client is ensured that the public key in the digital certificate indeed belongs to that domain name. The client then uses this public key to establish a shared session key (thus a secure connection) with the server and the server is authenticated by demonstrating the possession of the corresponding private key. Over this secure connection is the client’s password credential sent for client authentication.