MIS Review Vol. 16, No. 2, March (2011), pp. 93-110

Secure Password Based Authentication Protocol to Thwart Online Dictionary Attacks

Sandeep K. Sood
Department of Computer Science and Engineering, Guru Nanak Dev University Regional Campus, Gurdaspur, India

ABSTRACT: *Password is the most universally used authentication technique to authenticate the users on the web. Password based authentication protocols are vulnerable to dictionary attacks by means of automated programs because most of the user chosen passwords are limited to the user's personal domain. In this paper, we propose a secure password based authentication protocol in which the computation efforts required from the attacker during login on to the web server increases with each login failure. The web server stores the cookie on the client's computer if the legitimate client authenticates itself to the web server. There after, the legitimate client can easily authenticate itself to the web server from a computer that contains cookie. However, the legitimate client or the attacker has to put up some additional computational efforts during login from a computer that does not contain cookie. The client generated dynamic authentication information is different for the same user in different sessions of Secure Socket Layer (SSL) protocol. The concept used in this paper is to combine traditional password authentication with a challenge that is easy to answer by the legitimate client and the computational cost of authentication increases for an attacker with each login failure. Therefore, even the automated programs can not launch online dictionary attacks on the proposed protocol. This protocol provides better protection against different types of attacks launched by the attacker. The proposed protocol is easy to implement and it removes the some of the deficiencies of previously suggested password based authentication protocols.*

KEYWORDS: *Hyper Text Transfer Protocol, Cookies, Password, Secure Socket Layer, Online Dictionary Attacks.*

1. Introduction

Hyper Text Transfer Protocol (HTTP) is used to provide interaction between the web browser and the web server. It is stateless because the HTTP server treats each request independent of any previous request from the same client. The HTTP server does not maintain the correlation of the user visits from the same browser between successive sessions. The users are always strange to the web server if the web server does not maintain the state and continuity of the user (Park and Sandhu, 2000). Statelessness on the web makes it difficult to carry out online financial transactions in e-commerce.

The merchant web server can not remember the users on the web server without a state mechanism. Therefore, the web server uses cookies to maintain the state and connection of the user with the web server. Cookie technology is the most innovative feature that made the web stateful. A number of the web applications built on the top of HTTP needs to be stateful and require cookies to maintain the user's state.

The web server creates a cookie that contains the state information of a client and stores it on the client computer from where the request originated. The web server uses cookies to authenticate HTTP requests from the same client and to maintain persistent client state. Cookie enabled server can maintain information related to the client that can be used by the server during subsequent login request from the same client. The client's browser attaches the cookie with each subsequent request made by the client to the same web server. The web server retrieves the user's information from this cookie. The default parameters of HTTP cookie are cookie name, value, expiration date, URL path for which the cookie is valid, domain name and a flag to indicate whether the cookie had been sent using the SSL protocol. Secure cookies are required so that they can not be forged and all of their contents are not readable (Park and Sandhu, 2000; Liu et al., 2005). These secure cookies use different cryptographic primitives such as message digest, message authentication code, digital signature and encryption.

Cookies strengthen the connections between the legitimate client and the genuine web server across the web. It helps the web server to keep track of the user's movement and his behavior on the visited web server. Therefore, the web server can obtain significant information about the long term habits of their clients. There is no notification mechanism to alert the users when the cookies are being placed on their computer. The users are not aware of what information about them is being stored in the cookies. Cookies can persist for many years like google search engine routinely sets an expiration date in the year 2038 for its cookies. Third party cookies can be used by the online business organizations to create detailed records on the user's web browsing habits. Cookies can be used in conjunction with passwords to provide different levels of authentication to the users.

Password is the most commonly used authentication technique to authenticate the users on the web. Short and easily memorable passwords are susceptible to different attacks such as dictionary, phishing, stolen verifier, man-in-the-middle and insider attacks. On the other hand, the users find it difficult to memorize long and complex passwords. The concept of virtual password helps to defend the password authentication protocols from different types of attacks. Virtual password is a dynamic password that will be different for each new session between the same client and the server. The virtual password involves some computation on the client side to generate different password corresponding to the same user in different login sessions based on a single password shared between the client and the server (Lei et al., 2008).

The online dictionary attacks are the one of the major concerns in password based authentication protocols. A solution is required in which it is not possible for the attacker to launch online dictionary attack on password based authentication protocol. The aim of this paper is to provide a secure password based authentication solution using cookies for the user authentication. The main feature of the proposed protocol is that the legitimate client can easily login on to the web server. The computation complexity of authentication protocol for an attacker increases with each login failure. The protocol proposed in this paper is very effective and suitable to the business organizations such as online banks and online credit card organizations because the complexity of computation on the client side increases with each login failure so that the attacker can not impersonate as a legitimate user.

This paper is organized as follows. In Section 2, we explore the literature on the cookies and password based authentication protocols. In Section 3, we present the proposed password based authentication protocol. Section 4 discusses the security analysis of the proposed protocol. Section 5 concludes the paper.

2. Related work

Cookies are obscure to the users and are completely controlled by the web server. Therefore, cookies are good choice for a single sign-on (SSO) solution. Samar (1999) suggested SSO using HTTP cookies for web based environment. He suggested three approaches namely centralized cookie server, decentralized cookie server and centralized login server to provide SSO for web applications. The client can choose any of the three SSO solutions depending upon the requirements of web application in terms of deployability, performance and management.

Park and Sandhu (2000) suggested address based (IP_Cookie), password based (Pswd_Cookie) and digital signature based (Sign_Cookie) secure cookies for the user authentication. They suggested different set of inter dependent cookies such as name cookie, life cookie, password cookie and seal cookie. The role server issues one or more cookies by storing it on the client's computer. As the client connects to the web server, the relevant cookies are transmitted to the web server. Any of the web servers that accept these cookies verifies the cookie and provides the access of resources depending upon the role of the cookie. These secure cookies are used for user authentication especially in e-commerce transactions on the web.

Fu et al. (2001) designed a secure cookie based client authentication framework in conjunction with Secure Socket Layer (SSL) protocol based on informal survey of commercial protocols. They claimed that their protocol is secure against different

attacks launched by the attacker. Liu et al. (2005) analyzed and found that Fu et al.'s protocol is vulnerable to cookie replay, volume attacks and does not provide high-level confidentiality. Therefore, they proposed a cookie based authentication protocol that provides confidentiality, integrity and protection from replay attacks. Their scheme does not involve any database lookup or public key cryptography. It also does not require changes in Internet cookie specification and can be easily deployed on an existing web server.

Xu et al. (2002) presented a cookie based authentication protocol in which the server stores credit card information of each client in their respective cookie. They exploited the concept of secure distributed storage by storing some sensitive information in the HTTP cookie in encrypted form. The web server stores the One Time Pad (OTP) keys in its local database and encrypt/decrypt the cookies using these keys. This protocol can not handle multiple simultaneous requests with the same cookie. Moreover, the server has the overhead of encryption and decryption for verifying each cookie and also has to do database lookups.

Pinkas and Sander (2002) suggested Reverse Turing Tests (RTT) for authentication so that human user can easily pass out the test but it is very difficult for the automated program to pass out the test. Pinkas and Sander assumed that the users login from a limited set of computers containing activated cookies. The user is asked to pass RTT during login from a new computer or after entering a wrong password from a trusted computer. Stubblebine and Oorschot (2004) observed that RTT based protocols are vulnerable to RTT relay attacks. To counter these kinds of RTT relay attacks, Stubblebine and Oorschot developed a protocol based on the user's login history and suggesting modifications to Pinkas and Sander's RTT based protocol so that only trustworthy machines are used to store cookies.

Blundo, Cimato, and Prisco (2005) proposed encrypted cookies based web authentication protocol. The main weakness of this cookie based protocol is that the server has to do database lookups for verifying each received cookie. Wang et al. (2005) presented cookies based password authentication protocol that uses cryptographic puzzles to prevent online dictionary attacks. Their scheme increases the computational burden for an attacker and imposing negligible load on the legitimate clients as well as on the authentication server.

Juels, Jakobsson, and Jagatic (2006) suggested the use of cache cookies for the user identification and authentication that uses the browser cache files to identify the browser. These cookies are easy to deploy because it does not require installation of any software on the client side. Then they extended the concept to active cookie scheme, which stores the user's identification and a fixed IP address of the server. During the client's visits to

the server, the server will redirect the client request to the fixed IP address so as to defeat phishing and pharming attack. Goyal et al. (2006) proposed an authentication protocol that prevents online dictionary attacks and is easy to implement without any infrastructural changes. This protocol uses challenge response mechanism and one way hash function to thwart online dictionary attacks. The legitimate user can easily login on to the web server and the computational efforts increases for the attacker trying thousands of authentication requests in an attempt to launch online dictionary attack. Karlof et al. (2007) proposed the cookies based Locked Same Origin Policy (LSOP) that enforces access control for the SSL web objects based on the server's public key. Later on, LSOP is found to be susceptible to phishing attack.

Lei et al. (2008) proposed a virtual password concept based on the randomized linear functions involving a small amount of human computing to secure the user's password in online transaction. They analyzed that their scheme defends against phishing, key logger and shoulder surfing attacks. Wu et al. (2008) proposed SSO anti-phishing technique based on encrypted cookie that defeats phishing and pharming attacks. It encrypts the sensitive data with the server's public key and stores this cookie on the user's computer. This Encrypted Cookie Scheme (ECS) has advantage that the user can ignore SSL indicator in online transaction procedure. Microsoft's Passport (Window Live ID) initiative (Microsoft Passport November, 2009) is a cookie based password management system. This service authenticates the user to different web sites that are under the control of this centralized system. The main limitations of this approach are that the users have to trust the centralized server and it requires web administration changes on those sites that use this system for its authentication (Kormann and Rubin, 2000).

Sood et al. (2009) proposed a cookie based single password anti-phishing protocol that is secure against different possible attacks. In this protocol, the client machine's browser generates a dynamic identity and a dynamic password for each login request to the server. The dynamic identity and dynamic password will be different for the same client in different sessions of the SSL protocol. The proposed protocol makes financial transactions more secure on the web as it is practical and efficient. The client can use a single password for different online accounts and that password can not be detected by any of the malicious server or the attacker. The protocol is equally secure for security ignorant users, who are not very conversant with the browser's security indicators.

3. Proposed protocol

A HTTP cookie contains information related to the user such as user name, domain name and token for authentication. It is designed and created by the web server and stored

on the user's computer to keep track of the client state. The cookie is transferred back from the client's computer to the web server in succeeding login requests by the client. The cookies are server controlled hence the design and contents of a cookie are decided by the web server without requiring any infrastructural changes on the client side. The web server decides various fields required in the cookie depending upon the information that the web server wants to keep related to their clients.

The proposed scheme provides cookies based password authentication protocol for online password management. The legitimate client can easily login on to the web server using his identity and password. An attacker can not launch online dictionary attacks because computation efforts on the client side increases with each login failure. The proposed protocols run on top of the SSL protocol (Freier, Karlton, and Kocher, 1996) and comprise four phases as follows. The notations used in this section are listed in Table 1.

Table 1 Notations

U_i	i^{th} User
S	Server
ID_i	Unique Identification of User U_i
P_i	Password of User U_i
URL	Destination Web Site
H( )	One-Way Hash Function
MAX_TRUST	Maximum Trust Assigned to User U_i
MIN_TRUST	Minimum Trust Assigened to User U_i
CUR_TRUST	Cruuent Trust Value of User U_i
TRUST_BITS	To be Computed by User U_i or Guessed by Attacker
SK	Private Key of Server
PK	Public Key of Server
SS	Session Key
$\oplus$	XOR Operation
|	Concatenation

We present two authentication protocols. Each protocol has four phases. These two protocols have same registration phase and password change phase and they differ in login and authentication phase. Protocol 1 makes use of cookies for the user's authentication whereas Protocol 2 does not use cookies. The user U_i has to follow the Protocol 1 if the user U_i's computer contains cookie else the user U_i has to follow Protocol 2.

3.1. Protocol 1

This protocol is shown in Figure 1 and Figure 2 and its various phases are described below.

3.1.1 Registration phase

The user U_i registers with the web server S by submitting his identity ID_i and password P_i to the web server S over a secure communication channel established using SSL protocol.

Step 1: $U_i \rightarrow S: ID_i, P_i$

In this paper, the concept of trust (MAX_TRUST, MIN_TRUST and CUR_TRUST) has been used and is defined as "to have belief or confidence in the honesty, goodness, skill or safety of the legitimate user". The web server S computes $A_i = P_i \oplus SK \oplus H (SK | ID_i)$ and stores ID_i, A_i, MAX_TRUST, MIN_TRUST and CUR_TRUST in its database. The web server S can frame its trust management policies and accordingly assign trust values (MAX_TRUST, MIN_TRUST and CUR_TRUST) to different users. Suppose the web server S decides the fixed MAX_TRUST value that represents the maximum trust to be 20, fixed MIN_TRUST value that represents the minimum trust to be 0 and variable CUR_TRUST value that represents the current trust value assigned to the user U_i. Initially, the web server S sets CUR_TRUST value equal to MIN_TRUST value and hence initial CUR_TRUST value will be 0. The CUR_TRUST value stored in the database of web server S is incremented by one after each successful login attempt by the user U_i on the web server S and decremented by one on each login failure. Once the CUR_TRUST value stored on the web server becomes equal to MAX_TRUST, it is not incremented further even after successful login by the user U_i. After successive login failures, the CUR_TRUST value may become less than MIN_TRUST value.

The web server S chooses a random value N_s in such a way so that the value of cookie $CK = N_s$ must be unique for each client. The web server S stores CK corresponding to the user U_i's identity ID_i in its database and stores CK as cookie information along with public key PK on the client's computer when the user U_i authenticates itself successfully to the web server S. Once the cookie is expired, then the web server S chooses a new random value N_s' and computes fresh cookie $CK^{new} = N_s$' for the same client.

Step 2: $S \rightarrow U_i: CK, PK$

3.1.2. Login phase

The user U_i submits his identity ID_i and password P_i to the web browser. If the user U_i's computer contain cookie CK corresponding to the web server S then the user U_i

extracts public key PK from its cookie information. The user U_i's web browser generates a new session key (SS), encrypts it using the public key PK of the web server S as $(SS)_{PK}$, computes $K_i = H (ID_i \mid URL \mid PK \mid P_i \mid SS \mid CK)$ and sends all these parameters along with CK to the web server S as shown in Figure 1 and Figure 2.

Step 1: $U_i \rightarrow S: (SS)_{PK}, K_i, CK$

3.1.3. Authentication phase

The web server S decrypts the session key SS from $(SS)_{PK}$ using its private key SK. The web server S recognizes the user U_i from the received cookie CK and extracts ID_i, A_i, MAX_TRUST, MIN_TRUST and CUR_TRUST corresponding to cookie CK from its database. Then the web server S computes P_i as $P_i = A_i \oplus SK \oplus H (SK \mid ID_i)$, the dynamic authentication information $K_i' = H (ID_i \mid URL \mid PK \mid P_i \mid SS \mid CK)$ and verifies it with the received value of K_i. If both values are equal, the web server S proceeds to the next step. Otherwise, the login request from the user U_i is rejected.

Case 1:

If CUR_TRUST value is more than or equal to MIN_TRUST value then the web server S chooses random nonce value N_k, computes $Z_i = N_k \oplus H (ID_i \mid SS \mid P_i)$, $V_i = H (ID_i \mid P_i \mid SS \mid N_k)$ and sends Z_i, V_i along with its public key certificate to the user U_i's web browser. The user U_i's web browser verifies the authenticity of public key certificate of the web server S and then computes $N_k = Z_i \oplus H (ID_i \mid SS \mid P_i)$. Then the user U_i's web browser computes dynamic authentication information $V_i' = H (ID_i \mid P_i \mid SS \mid N_k)$ and verifies the computed value of V_i' with the received value of V_i to validate that the messages are sent by the legitimate server S and not tampered during transmission. This equivalency authenticates the legitimacy of the user U_i and the web server S and the login request is accepted else the connection is interrupted. Hence the mutual authentication between the client and the server is achieved as shown in Figure 1. If the CUR_TRUST value stored in the database of web server S is less than MAX_TRUST value then the CUR_TRUST value is incremented by one (CUR_TRUST = CUR_TRUST + 1) after successful login attempt by the user U_i on the web server S. Finally, the user U_i and the web server S agree on the common session key as $S_k = H (SS \mid ID_i \mid N_k \mid CK \mid P_i)$. Afterwards, all the subsequent messages between the user U_i and the web server S are encrypted with this session key. Therefore, either the user U_i or the web server S can retrieve the original message because both of them know the common session key. If the user U_i fails to authenticate itself to the web server S then the web server S decreases the CUR_TRUST value by one (CUR_TRUST = CUR_TRUST - 1) and the server S removes the cookie CK from the client's computer.

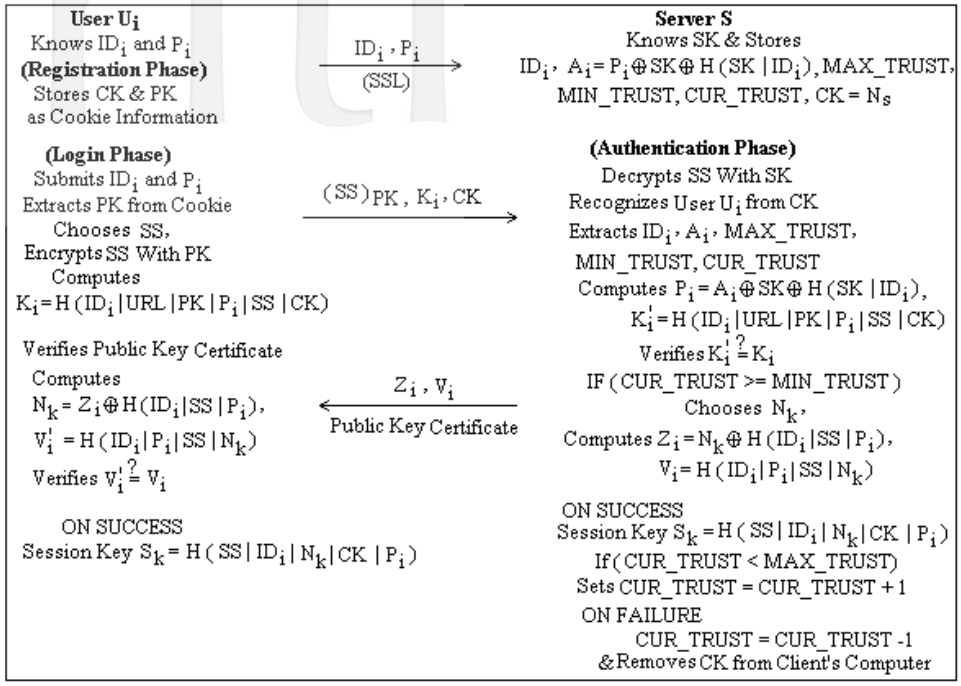

Figure 1 Protocol 1: (Case 1) Password Authentication Protocol with Cookie

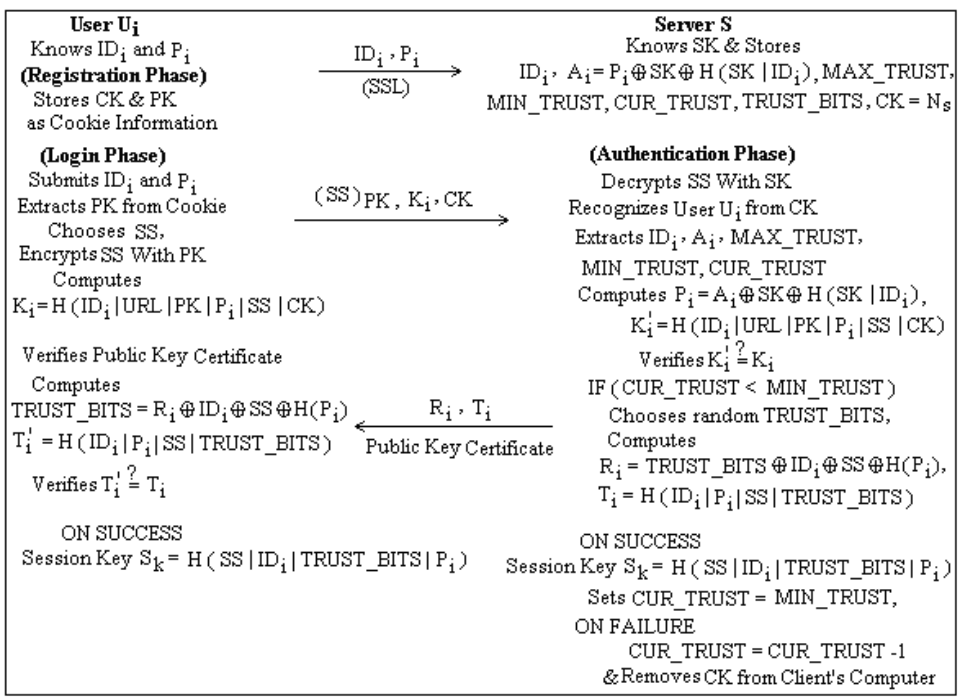

Figure 2 Protocol 1: (Case 2) Password Authentication Protocol with Cookie

Case 2:

If CUR_TRUST value is less than MIN_TRUST value then the web server S chooses random TRUST_BITS value having random number of bits. The web server S increases the number of bits in TRUST_BITS by one after each login failure. Suppose the number of bits in TRUST_BITS value is 3. That means if the user failed to login in this attempt then in next attempt the number of bits in TRUST_BITS value will be 4. Then the web server S computes R_i = TRUST_BITS $\oplus$ $ID_i \oplus SS \oplus H(P_i)$, $T_i = H(ID_i \mid P_i \mid SS \mid$ TRUST_BITS) and sends R_i, T_i along with its public key certificate to the user U_i's web browser. The user U_i's web browser verifies the authenticity of public key certificate of the web server S and then computes TRUST_BITS = $R_i \oplus ID_i \oplus SS \oplus H(P_i)$, $T_i' = H(ID_i \mid P_i \mid SS \mid$ TRUST_BITS) and verifies the computed value of T_i' with the received value of T_i. Hence the mutual authentication between the user U_i and the web server S is achieved as shown in Figure 2. Finally after successful authentication, the user U_i and the web server S agree on the common session key as $S_k = H(SS \mid ID_i \mid$ TRUST_BITS $\mid P_i)$. Afterwards, all the subsequent messages between the user U_i and the web server S are encrypted with this session key. Therefore, either the user U_i or the web server S can retrieve the original message because both of them know the common session key. Then the web server S resets the CUR_TRUST value equal to MIN_TRUST value after successful authentication. If the user U_i fails to authenticate itself to the web server S then the web server S decreases the CUR_TRUST value by one (CUR_TRUST = CUR_TRUST - 1) and the server S removes the cookie CK from the client's computer. On the other hand, the attacker has to guess the values of SS, ID_i, TRUST_BITS and P_i to compute the common session key as $S_k = H(SS \mid ID_i \mid$ TRUST_BITS $\mid P_i)$. The computational efforts required by the attacker to find the TRUST_BITS value increases exponentially with each login failure because the number of bits in TRUST_BITS increases by one after each login failure.

3.1.4 Password change protocol

The legitimate user U_i authenticates itself to the web server S using the Protocol 1 or Protocol 2. Once the mutual authentication between the user U_i and the web server S is achieved, the user U_i submits $Y_i = SS \oplus P_i \oplus P_i^{new}$ and $X_i = H(ID_i \mid P_i \mid SS \mid P_i^{new})$ to the web server S. The web server S retrieves P_i^{new} from Y_i as $P_i^{new} = Y_i \oplus SS \oplus P_i$, computes $X_i^* = H(ID_i \mid P_i \mid SS \mid P_i^{new})$ and verifies the computed value of X_i^* with the received value of X_i to validate that the messages are sent by the legitimate user U_i and not tampered during transmission. Afterwards, the web server S updates the value of $A_i = P_i \oplus SK \oplus H(SK \mid ID_i)$ stored in its database with $A_i^{new} = P_i^{new} \oplus SK \oplus H(SK \mid ID_i)$ and the password gets changed.

3.2 Protocol 2

This protocol is shown in Figure 3 and its various phases are described below.

3.2.1 Registration phase

The registration phase is same as in Protocol 1. (See Section 3.1)

3.2.2 Login phase

The user U_i submits his identity ID_i and password P_i to the web browser. If the user U_i's computer does not contain cookie CK then the user U_i's web browser establishes a connection with the web server S using the SSL protocol. In the SSL protocol, the web server S authenticates itself to the user U_i with its public key certificate. Then the user U_i generates a new SSL session key (SS), encrypts it using the public key PK of the web server S as $(SS)_{PK}$. Then the web browser chooses random nonce value N_r, computes $B_i = N_r \oplus H(P_i)$, $C_i = ID_i \oplus SS$ and $D_i = H(ID_i \mid URL \mid SS \mid P_i \mid N_r)$. The web browser of user U_i submits $(SS)_{PK}$, B_i, C_i and D_i to the web server S as shown in Figure 3. The web server S decrypts the SSL session key SS from $(SS)_{PK}$ using its private key SK. Then all the subsequent messages of this protocol are transmitted in insecure communication channel like Internet without using SSL protocol.

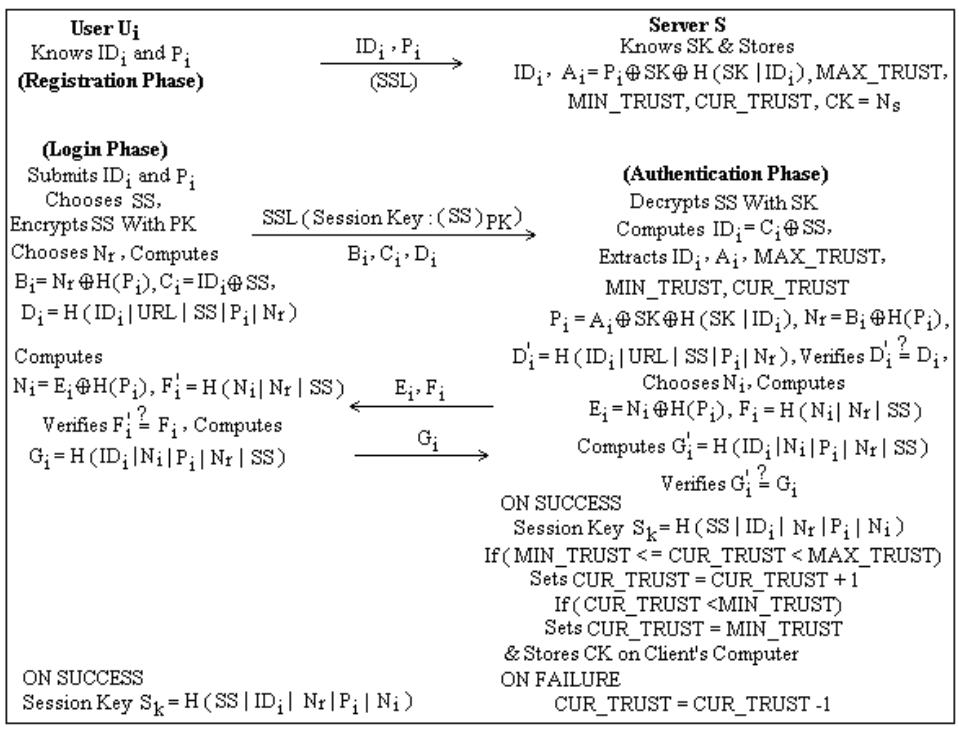

Figure 3 Protocol 2: Password Authentication Protocol without Cookie

3.2.3 Authentication phase

The web server S computes ID_i from C_i as $ID_i = C_i \oplus SS$ and recognizes the user U_i from its identity ID_i. Then the web server S extracts A_i, MAX_TRUST, MIN_TRUST and CUR_TRUST corresponding to user U_i's identity ID_i from its database. After that, the web server S computes P_i as $P_i = A_i \oplus SK \oplus H (SK \mid ID_i)$ and N_r from B_i as $N_r = B_i \oplus H (P_i)$. Afterwards, the web server S computes $D_i' = H (ID_i \mid URL \mid SS \mid P_i \mid N_r)$ and verifies it with the received value of D_i. If both values are equal, the web server S proceeds to the next step. Otherwise, the login request from the user U_i is rejected. The web server S chooses random nonce value N_i and computes $E_i = N_i \oplus H (P_i)$, $F_i = H (N_i \mid N_r \mid SS)$ and sends E_i and F_i to the web browser of user U_i. The web browser computes N_i from E_i as $N_i = E_i \oplus H (P_i)$ because the web browser knows password P_i of the user U_i. Then the web browser computes $F_i' = H (N_i \mid N_r \mid SS)$ and verifies the computed value of F_i' with the received value of F_i to validate that the messages are sent by the legitimate server S and not tampered during transmission. Then the user U_i's web browser computes dynamic authentication information $G_i = H (ID_i \mid N_i \mid P_i \mid N_r \mid SS)$ and sends it to the web server S. The web server S computes $G_i' = H (ID_i \mid N_i \mid P_i \mid N_r \mid SS)$ and compares the computed value of G_i' with the received value of G_i. This equivalency authenticates the legitimacy of the user U_i and the web server S and the login request is accepted else the connection is interrupted. Hence the mutual authentication between the user U_i and the web server S is achieved as shown in Figure 3. Afterwards, the web server S checks CUR_TRUST value in its database corresponding to the user identity ID_i. If CUR_TRUST value stored in its database is more than or equal to MIN_TRUST but less than MAX_TRUST then the web server increases the CUR_TRUST value by one (CUR_TRUST = CUR_TRUST + 1) after successful authentication. If CUR_TRUST value stored in its database is less than MIN_TRUST then the web server resets the CUR_TRUST value equal to MIN_TRUST value after successful authentication. If the user U_i fails to authenticate itself to the web server S then the web server S decreases the CUR_TRUST value by one (CUR_TRUST = CUR_TRUST - 1). After successful authentication, the user U_i and the web server S agree on the common session key as $S_k = H (SS \mid ID_i \mid N_r \mid P_i \mid N_i)$ and stores the cookie CK on the client's computer. Afterwards, all the subsequent messages between the user U_i and the web server S are encrypted with this session key. Therefore, either the user U_i or the web server S can retrieve the original message because both of them know the common session key.

3.2.4 Password change phase

The password change phase is same as in Protocol 1. (See Section 3.1)

4. Security analysis

The security of messages in online transaction inside communication channel is managed with SSL protocol. The proposed cookies based password authentication protocol uses SSL protocol to establish SSL session key (SS) and then all the succeeding messages are communicated without using SSL protocol. This protocol provides good protection especially against online dictionary attacks. A good password authentication protocol should provide protection from different feasible attacks.

1. **Online dictionary attack:** In this type of attack, the attacker pretends to be legitimate client and attempts to login on to the server by guessing different words as password from a dictionary. In the proposed protocol, the attacker has to generate $K_i = H (ID_i | URL | PK | P_i | SS | CK)$ or $\{B_i = N_r \oplus H (P_i), C_i = ID_i \oplus SS$ and $D_i = H (ID_i | URL | SS | P_i | N_r)\}$ corresponding to the user U_i, which is different for each new SSL session. With each failed login attempt, the difficulty of guessing TRUST_BITS value increases because number of bits increases by one in TRUST_BITS value after each login failure and sooner the guessing of TRUST_BITS value will go out of the scope of the attacker as shown in Figure 2 (case 2). Moreover, the attacker has to guess ID_i, N_i, P_i, N_r and SS correctly at the same time to compute the dynamic identity $D_i = H (ID_i | URL | SS | P_i | N_r)$ and dynamic authentication information $G_i = H (ID_i | N_i | P_i | N_r | SS)$ as shown in Figure 3. The legitimate user U_i can easily login on to the web server S, whatever may be the CUR_TRUST value. Therefore, the proposed protocol is secure against online dictionary attack.

2. **Offline dictionary attack:** In offline dictionary attack, the attacker can record messages and attempts to guess the user's identity and password from the recorded messages. The attacker obtains some identity and password verification information such as $K_i = H (ID_i | URL | PK | P_i | SS | CK)$ or $\{Z_i = N_k \oplus H (ID_i | SS | P_i)$ and $V_i = H (ID_i | P_i | SS | N_k)\}$ or $\{R_i = TRUST\_BITS \oplus ID_i \oplus SS \oplus H (P_i)$ and $T_i = H (ID_i | P_i | SS | TRUST\_BITS)\}$ or $\{B_i = N_r \oplus H (P_i), C_i = ID_i \oplus SS$ and $D_i = H (ID_i | URL | SS | P_i | N_r)\}$ or $\{E_i = N_i \oplus H (P_i), F_i = H (N_i | N_r | SS)$ and $G_i = H (ID_i | N_i | P_i | N_r | SS)\}$. The attacker can not compute ID_i and P_i from these recorded messages. Therefore, the proposed protocol is secure against offline dictionary attack.

3. **Eavesdropping attack:** In this type of attack, the attacker first listens to all the communications between the client and the server and then tries to find out the client's identity ID_i and password P_i. The client's browser uses random nonce value N_r and SSL session key SS for the generation of dynamic identity and password verifier information $\{B_i = N_r \oplus H (P_i), C_i = ID_i \oplus SS$ and $D_i = H (ID_i | URL | SS | P_i | N_r)\}$ or $K_i = H (ID_i | URL | PK | P_i | SS | CK)$ corresponding to the user U_i, which is

different for each new SSL session. Also, the eavesdropper can not compute the user U_i's identity ID_i and password P_i from any of the recorded message. Therefore, the proposed protocol is secure against eavesdropping attack.

4. **Denial of service attack:** In a specific type of denial of service attack, the server is cheated by the attacker to update the password verifier information with some false password verification information so that the legitimate user can not login successfully in subsequent login request to the server. The user U_i can change his password after the client and the server authenticate each other using the protocol shown in Figure 1 or Figure 2 or Figure 3. Therefore, the proposed protocol is secure against the user specific denial of service attack.

5. **Phishing attack:** In this type of attack, the attacker sends spoofed e-mails to different users from a web site that is under the control of the attacker. Victim enters his valid login credentials into the fraudulent web site that allows the attacker to transfer funds from the victim's account or cause other damages. The proposed protocol generates a new dynamic identity and password verifier information $\{B_i = N_r \oplus H(P_i)$, $C_i = ID_i \oplus SS$ and $D_i = H(ID_i \mid URL \mid SS \mid P_i \mid N_r)\}$ or $K_i = H(ID_i \mid URL \mid PK \mid P_i \mid SS \mid CK)$ corresponding to the user U_i, which is different for each new SSL session. The fraudulent server can ignore dynamic identity and password verifier information but can not produce valid credentials $\{Z_i = N_k \oplus H(ID_i \mid SS \mid P_i)$ and $V_i = H(ID_i \mid P_i \mid SS \mid N_k)\}$ or $\{R_i = TRUST\_BITS \oplus ID_i \oplus SS \oplus H(P_i)$ and $T_i = H(ID_i \mid P_i \mid SS \mid TRUST\_BITS)\}$ or $\{E_i = N_i \oplus H(P_i)$ and $F_i = H(N_i \mid N_r \mid SS)\}$ meant for the user U_i because it does not have any such credentials. Therefore, the proposed protocol is secure against phishing attack.

6. **Pharming attack:** Pharming is a technique that fools the user by connecting his machine to a fake web site even when the user submits correct domain name in to the web browser. This technique exploits vulnerabilities in the DNS servers to distribute the fake address information by DNS spoofing attack. Like phishing attacks, the attacker sets up a capture site to collect identity and password verifier information. The attacker can cause the DNS caching server to return false information and direct the user to a malicious site. Malicious site can not impersonate as valid server because it can not generate valid credentials $\{Z_i = N_k \oplus H(ID_i \mid SS \mid P_i)$ and $V_i = H(ID_i \mid P_i \mid SS \mid N_k)\}$ or $\{R_i = TRUST\_BITS \oplus ID_i \oplus SS \oplus H(P_i)$ and $T_i = H(ID_i \mid P_i \mid SS \mid TRUST\_BITS)\}$ or $\{E_i = N_i \oplus H(P_i)$ and $F_i = H(N_i \mid N_r \mid SS)\}$ meant for the user U_i, which are unique for each new session. Therefore, the attacker can not launch pharming attack on the proposed protocol.

7. **Man-in-the-middle attack:** In this type of attack, the attacker intercepts the messages sent between the client and the server and replay these intercepted messages with in

the valid time frame window. The attacker can act as the client to the server or vice-versa with recorded messages. In the proposed protocol, the attacker can intercept the login request message $\{B_i = N_r \oplus H(P_i)$, $C_i = ID_i \oplus SS$ and $D_i = H(ID_i \mid URL \mid SS \mid P_i \mid N_r)\}$ or $K_i = H(ID_i \mid URL \mid PK \mid P_i \mid SS \mid CK)$ corresponding to the user U_i, which is sent by a user U_i to the server S. Then he starts a new session with the server S by sending a login request by replaying the login request message with in the valid time frame window. The attacker can authenticate itself to server S as well as to legitimate user U_i but can not compute the session key $S_k = H(SS \mid ID_i \mid N_k \mid CK \mid P_i)$ or $S_k = H(SS \mid ID_i \mid TRUST\_BITS \mid P_i)$ or $S_k = H(SS \mid ID_i \mid N_r \mid P_i \mid N_i)$ because the attacker does not know the values of ID_i, P_i, SS, N_k, N_i, N_r and TRUST_BITS. Therefore, the proposed protocol is secure against man-in-the-middle attack.

8. **Replay attack:** In this type of attack, the attacker first listens to the communication between the client and the server, then tries to imitate the user to login on to the server by resending the captured messages. Replaying a message of one SSL session into another SSL session is useless because each SSL session generates a different dynamic identity and password verifier information corresponding to the same client because the session key SS is different for each new SSL session and hence messages can not be replayed successfully in any other SSL session. Moreover, the attacker can not compute the session key. Therefore, the proposed protocol is secure against message replay attack.

9. **Leak of verifier attack:** In this type of attack, the attacker may be able to steal verification table from the server. In case the password verifier information ID_i, $A_i = P_i \oplus SK \oplus H(SK \mid ID_i)$, MAX_TRUST, MIN_TRUST, CUR_TRUST, $CK = N_s$ is stolen by breaking into the server's database, the attacker does not have sufficient information to calculate the user's identity ID_i and password P_i because the attacker has to guess SK, ID_i and P_i correctly at the same time. It is not possible to guess all these parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against leak of verifier attack.

10. **Message modification or insertion attack:** In this type of attack, the attacker modifies or inserts some messages on the communication channel with the hope of discovering the client's password or gaining unauthorized access. Modifying or inserting messages in the proposed protocol can result in authentication failure between the client and the server but can not allow the attacker to gain any information about the client's password or gain unauthorized access. Therefore, the proposed protocol is secure against message modification or insertion attack.

11. **Brute force attack:** To launch brute force attack, an attacker first obtains some password verification information such as $\{K_i = H(ID_i \mid URL \mid PK \mid P_i \mid SS \mid CK)\}$

from Figure 1 or Figure 2 protocol or {$B_i = N_r \oplus H(P_i)$, $C_i = ID_i \oplus SS$ and $D_i = H(ID_i \mid URL \mid SS \mid P_i \mid N_r)$} from Figure 3 protocol. Even after recording these messages, the attacker has to guess out minimum two parameters out of ID_i, P_i, N_r and SS correctly at the same time. It is not possible to guess out two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against brute force attack.

5. Conclusion

Password based authentication protocols are mainly vulnerable to dictionary attacks. Therefore, password theft is growing significantly and frightening the confidence of customer in e-commerce. Transaction authorization method based on out of band channels like SMS messages are introduced by banks to thwart dictionary and phishing attacks but it requires two separate communication channels for the user's authentication. We have specified and analyzed a cookie based password authentication protocol which is very effective to thwart online dictionary attacks because the computation cost of login on to the web server increases exponentially with each login failure for an attacker. The legitimate client can easily authenticate itself to the web server from any computer irrespective of whether that computer contains cookie or not. The proposed protocol is simple and fast if the user is using valid identity and correct password for its authentication. This protocol is practical and efficient because only one-way hash functions and XOR operations are used in its implementation. Security analysis proved that the proposed protocol is secure and practical.

References

1. Blundo, C., Cimato S. and Prisco, R.D. (2005) 'A lightweight approach to authenticated web caching', *Proceedings IEEE International Symposium on Applications and the Internet (SAINT 2005)*, Trento, Italy, pp. 157-163.

2. Freier, A.O., Karlton P. and Kocher, P.C. (1996) 'The SSL protocol version 3.0', Internet draft, http://tools.ietf.org/html/draft-ietf-tls-ssl-version3-00.

3. Fu, K., Sit, E., Smith K. and Feamster, N. (2001) 'Dos and don'ts of client authentication on the web', *Proceedings 10th USENIX Security Symposium*, Washington, DC, pp. 1-16.

4. Goyal, V., Kumar, V., Singh, M., Abraham A. and Sanyal, S. (2006) 'A new protocol to counter online dictionary attacks', *Computers & Security*, Vol. 25, No. 2, pp. 114-120.

5. Juels, A., Jakobsson M. and Jagatic, T.N. (2006) 'Cache cookies for browser authentication', *Proceedings IEEE Symposium on Security and Privacy*, Oakland, CA, pp. 301-305.

6. Karlof, C., Shankar, U., Tygar J.D. and Wagner, D. (2007) 'Dynamic pharming attacks and locked same origin policies for web browsers', *Proceedings of the 14th ACM Conference on Computer and Communications Security*, New York, pp. 58-71.

7. Kormann, D.P. and Rubin, A.D. (2000) 'Risks of the passport single sign-on protocol', *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Vol. 33, No. 1, pp. 51-58.

8. Lei, M., Xiao, Y., Vrbsky, S.V. and Li, C.C. (2008) 'Virtual password using random linear functions for on-line services. ATM machines, and pervasive computing', *Computer Communications*, Vol. 31, No. 18, pp. 4367-4375.

9. Liu, A.X., Kovacs, J.M., Huang, C.T. and Gouda, M.G. (2005) 'A secure cookie protocol', *Proceedings 14th IEEE International Conference on Computer Communications and Networks*, San Diego, CA, pp. 333-338.

10. Microsoft Passport, accessed November 10, 2009, http://www.passport.net/

11. Park, J.S. and Sandhu, R. (2000) 'Secure cookies on the web', *IEEE Internet Computing*, Vol. 4, No. 4, pp. 36-44.

12. Pinkas, B. and Sander, T. (2002) 'Securing passwords against dictionary attacks', *Proceedings 9th ACM Conference on Computer and Communication Security*, New York, pp. 161-170.

13. Samar, V. (1999) 'Single sign-on using cookies for web applications', *Proceedings 8th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises*, Washington, DC, pp. 158-163.

14. Sood, S.K., Sarje A.K. and Singh, K. (2010) 'Dynamic identity based single password anti-phishing protocol', *Security and Communication Networks*, John Wiley, doi.wiley.com/10.1002/sec.169

15. Stubblebine S.G. and Oorschot, P.C.V. (2004) 'Addressing online dictionary attacks with login histories and humans in the loop', *Lecture Notes in Computer Science (LNCS)*, Vol. 3110, pp. 39-53.

16. Wang, P., Kim, Y., Kher V. and Kwon, T. (2005) 'Strengthening password-based authentication protocols against online dictionary attacks', *Lecture Notes in Computer Science (LNCS)*, Vol. 3531, pp. 17-32.

17. Wu, Y., Yao H. and Bao, F. (2008) 'Minimizing SSO effort in verifying SSL anti-phishing indicators', *International Federation for Information Processing (IFIP)*, Vol. 278, pp. 47-61.

18. Xu, D., Lu, C. and Santos, A.D. (2002) 'Protecting web usage of credit cards using one-time pad cookie encryption', *Proceedings 18th Annual Computer Security Applications Conference*, Washington, DC, pp. 51-58.

About the author

Sandeep Kumar Sood received his M.Tech (Computer Science & Engineering) in 1999 from the Guru Jambheshwar University Hisar (Haryana), India. He has completed his PhD in 2010 from Department of Electronics and Computer Engineering at Indian Institute of Technology, Roorkee (Uttrakhand), India. He is working as a lecturer (Computer Science & Engineering) in Guru Nanak Dev University Regional Campus, Gurdaspur (Punjab), India. His research interests include Authentication Protocols, Computer and Network Security, Cryptography and Computer Networks.