Information Security Journal: A Global Perspective
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t768221795

Cookie-Based Virtual Password Authentication Protocol
Sandeep K. Sood*
* Department of Computer Science and Engineering, G.N.D.U. Regional Campus, Gurdaspur, India

Online publication date: 11 April 2011

To link to this Article DOI: 10.1080/19393555.2011.560924
URL: http://dx.doi.org/10.1080/19393555.2011.560924

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.informaworld.com/terms-and-conditions-of-access.pdf

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.
ABSTRACT  The password is the most common technique used to authenticate Web users. Password-based authentication protocols are susceptible to automated dictionary attacks because most passwords are chosen by users from their personal domain. In this paper, we propose a cookie-based virtual password authentication protocol that preserves the advantages of conventional password authentication while simultaneously increasing the efforts required for online dictionary attacks. The Web server stores the cookie on the user's machine if the legitimate user authenticates to the Web server. Thereafter, the legitimate user can easily authenticate to the Web server from a machine that contains the cookie. However, the legitimate user requires some additional computational efforts during login from a machine that does not contain the cookie. The computation efforts required from the attacker during login to the Web server increases exponentially with each login failure. The user generated virtual password 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 user but computational cost increases for the attacker with each login failure. Therefore, even the automated programs cannot launch online dictionary attacks on the proposed protocol. This protocol provides good security against different types of attacks launched by the attacker. The proposed protocol is easy to implement and removes some of the drawbacks of earlier proposed password-based authentication protocols.

KEYWORDS  cookies, hyper text transfer protocol, virtual password, secure socket layer, online dictionary attacks

1. INTRODUCTION

The Web browser uses hyper text transfer protocol (HTTP) for interaction with the Web server. HTTP does not maintain the correlation of the user visits from the same browser to the same Web server between successive sessions; hence, it is stateless. The users are always unknown to the Web server if it does not maintain the state and association of the user (Park & Sandhu, 2000).
Statelessness on the Web makes it difficult to carry out online financial transactions in e-commerce. The Web server cannot remember users without a state mechanism. Therefore, the Web server uses cookies to maintain the state and association of the user with the Web server. Cookie technology is the most innovative feature that made the Web stateful. A number of Web applications built on HTTP need to be stateful and need cookies to maintain the user’s state.

The Web server creates a cookie that contains the state information of a user and stores it on the user’s machine from where the request is generated. The Web server uses cookies to authenticate HTTP requests from the same user and to maintain stable user state. Cookie-enabled servers can maintain information related to the user that can be used by the server during subsequent login request from the same user. The user’s browser attaches the cookie with each subsequent request made by the user 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 SSL protocol.

Secure cookies provide a good mean of authentication because they cannot be forged and all of their contents are not readable (Park & Sandhu, 2000; Liu, Kovacs, Huang, & Gouda, 2005). These secure cookies use different cryptographic primitives such as encryption, message authentication code (MAC), message digest and digital signature. Encryption is used to protect the data. It transforms plaintext using a cipher algorithm to make it unreadable to anyone except those possessing a secret key. MAC is a short piece of information used to authenticate a message. A MAC algorithm accepts secret key and an arbitrary-length message to be authenticated as an input. Then it outputs a MAC. This MAC value is verification information that protects both data integrity as well as authenticity of the message. This verification information is verified by the user who also possesses the secret key to detect any changes to the message content. Message digest also takes an arbitrary block of data as input and returns a fixed-size bit string cryptographic hash value. An accidental or intentional change to the data will result in mismatch of hash values computed by the sender and the receiver. It is different from the MAC in the sense that it does not use secret key for encoding of verification information.

A digital signature is a mathematical concept based on asymmetric cryptography. It is used for verifying the authenticity of a digital message or document. A valid digital signature gives a recipient to believe that the message is created by a known sender and has not been altered in transit. Digital signatures are commonly used for software distribution, financial transactions and in other cases where it is important to detect forgery and tampering.

Cookies bound the connections between the legitimate user and the Web server across the Web. It helps the Web server to keep track of the user’s behavior and state. Therefore, the Web server can obtain significant information about the long-term habits of their users. Cookies can persist for many years; for example, the 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.

The password is the most common technique used to authenticate Web users. Short and easily memorized passwords are susceptible to dictionary attack. On the other hand, 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 user and the server. Each new session is started whenever the user submits its identity and password verifier information for its authentication to the server. The virtual password involves some computation on the user side to generate different password corresponding to the same user in different login sessions based on a single password shared between the user and the server (Lei, Xiao, Vrbsky, & Li, 2008).

In an online dictionary attack, the attacker verifies the guessed identity and password verifier information on the server by submitting it to the server. The online dictionary attacks are one of current research area in password-based authentication protocols. A solution is required in which it is not possible for the attacker to launch an online dictionary attack on password-based authentication protocol. The aim of this paper is to provide a virtual password-based authentication solution using cookies for the user authentication. The main feature of the proposed protocol is that the legitimate user can easily log in to the Web server. The

Cookie-based Virtual Password Authentication Protocol
computation complexity of this authentication protocol for an attacker increases with each login failure. The protocol proposed in this paper is effective and suitable to the online transactions because the complexity of computation on the user side increases with each login failure so that the attacker cannot impersonate a legitimate user.

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

2. RELATED WORK

Users are not aware of what information about them is being stored in the cookies. These cookies 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 a 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 user can choose any of the three SSO solutions depending upon the requirements of the 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 user authentication. They suggested different set of interdependent cookies such as name cookie, life cookie, password cookie and seal cookie. The role server issues one or more cookies by storing them on the user’s machine. As the user 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 access to resources depending upon the role of the cookie. These secure cookies are used for user authentication especially in e-commerce transactions.

Fu, Sit, Smith, and Feamster (2001) designed a secure cookie-based user authentication framework in conjunction with secure socket layer (SSL) protocol based on an informal survey of commercial protocols. They claim that their protocol is secure against different attacks launched by the attacker. Liu, Kovacs, Huang, and Gouda (2005) analyzed and found that Fu et al.’s protocol is susceptible to cookie replay and 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, Lu, and Santos (2002) presented a cookie-based authentication protocol in which the server stores credit card information of each user in his or her 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 encrypts/decrypts the cookies using these keys. This protocol cannot 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 users can easily pass the test, but it is difficult for the automated program to pass the test. Pinkas and Sander assumed that the users log in from a limited set of machines containing activated cookies. The user is asked to pass RTT during login from a new machine or after entering a wrong password from a trusted machine. 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 (2004) 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 cookie-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, Kim, Kher, and Kwon (2005) presented cookie-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 users as well as on the authentication server.
Juels, Jakobsson, and Jagatic (2006) suggested the use of cache cookies for user identification and authentication that uses the browser cache files to identify the browser. These cookies are easy to deploy because they do not require installation of any software on the user side. Juels et al. (2006) extended the concept to active cookie scheme, which stores the user’s identification and a fixed IP address of the server. During the user’s visits to the server, the server will redirect the user 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 log in to the Web server, and the computational efforts increases for the attacker trying thousands of authentication requests in an attempt to launch an online dictionary attack. Karlof, Shankar, Tygar, and Wagner (2007) proposed the cookie-based Locked Same Origin Policy (LSOP) that enforces access control for SSL Web objects based on the server’s public key. Later, 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, Yao, and Bao (2008) proposed a single sign-on (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 machine. This encrypted cookie scheme (ECS) has an advantage that the user can ignore the SSL indicator in online transaction procedure. Microsoft’s Passport (Window Live ID) initiative (2009) is a cookie-based password management system. This service authenticates the user to different Websites 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 & Rubin, 2000).

Sood, Sarje, and Singh (2009) proposed a cookie-based single password anti-phishing protocol that is secure against different possible attacks. In this protocol, the user 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 user 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 user can use a single password for different online accounts, and that password cannot be detected by any of the malicious servers or attackers. The protocol is equally secure for security ignorant users, who are not very conversant with the browser’s security indicators. Sood et al. (2011) also proposed a dynamic identity-based authentication protocol for multiserver architecture. Sood et al. (2011) proposed an inverse cookie-based virtual authentication protocol in which the cookies are not being stored on the trustworthy machines; instead, the cookies are being stored on those machines from where the user failed to log in.

A secure dynamic identity and dynamic password-based authentication protocol is required so that attacker finds it difficult to launch an online dictionary attack. Moreover, the cost of computation of login message should increase exponentially for an attacker with each login failure.

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 machine to keep track of the user state. The cookie is transferred back from the user’s machine to the Web server in succeeding login requests by the user. The cookies are server controlled; hence, the design and contents of a cookie are decided by the Web server without requiring any infrastructure changes on the user 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 users.

The proposed scheme provides cookie-based virtual password authentication protocol for online password management. The legitimate user can easily log in to the Web server using his identity and password. An attacker cannot launch online dictionary attacks because computation efforts for the attacker on the user side increases exponentially with each login.
TABLE 1  Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U_i</td>
<td>_i_th User</td>
</tr>
<tr>
<td>S</td>
<td>Server</td>
</tr>
<tr>
<td>ID_i</td>
<td>Unique Identification of User U_i</td>
</tr>
<tr>
<td>P_i</td>
<td>Password of User U_i</td>
</tr>
<tr>
<td>MOB_NO</td>
<td>Mobile Phone Number of User U_i</td>
</tr>
<tr>
<td>URL</td>
<td>Destination Web Site</td>
</tr>
<tr>
<td>OTP</td>
<td>One Time Password of Server for Each Client</td>
</tr>
<tr>
<td>H()</td>
<td>One-Way Hash Function</td>
</tr>
<tr>
<td>MAX_TRUST</td>
<td>Maximum Trust Assigned to User U_i</td>
</tr>
<tr>
<td>MIN_TRUST</td>
<td>Minimum Trust Assigned to User U_i</td>
</tr>
<tr>
<td>CUR_TRUST</td>
<td>Current Trust Value of User U_i</td>
</tr>
<tr>
<td>TRUST_BITS</td>
<td>To be computed by User U_i or guessed by attacker</td>
</tr>
<tr>
<td>SK</td>
<td>Private Key of Server</td>
</tr>
<tr>
<td>PK</td>
<td>Public Key of Server</td>
</tr>
<tr>
<td>SS</td>
<td>Session Key of SSL Protocol</td>
</tr>
<tr>
<td>⊕</td>
<td>XOR Operation</td>
</tr>
<tr>
<td></td>
<td>Concatenation</td>
</tr>
</tbody>
</table>

failure. The notations used in this section are listed in Table 1.

The proposed protocol runs on top of the SSL protocol (Freier, Karlton, & Kocher, 1996) and has four phases. During the registration phase, the Web server stores a cookie on the user’s machine. However, the user is not restricted to log in only from this machine. The login phase will differ in case the user machine does not contain the cookie. Similarly the authentication phase will be different depending on whether the user’s machine contains the cookie or not.

### 3.1. Registration Phase

A new user has to register on the Web server S to become a legitimate client C. The request for registration by user U_i is sent by a registered e-mail address that contains the mobile number of the user U_i. Two temporary keys (TK_1 and TK_2) are generated by the server S and stored on its temporary database corresponding to e-mail and mobile number (MOB_NO). TK_1 is sent to the user U_i on his e-mail address and TK_2 is sent to the user U_i on his mobile number using short message service (SMS). Then the user U_i submits his identity ID_i, password P_i and mobile phone number MOB_NO to the Web server S over a secure communication channel established using SSL protocol as shown in Step 1.

Step 1: U_i → S: ID_i ⊕ TK_1 ⊕ TK_2, P_i ⊕ TK_1 ⊕ TK_2, MOB_NO.

The Web server S retrieves the value ID_i and P_i because it knows the value of TK_1 and TK_2 corresponding to MOB_NO of user U_i. Then it chooses random OTP for each user and stores ID_i, T_i = OTP ⊕ H(SK), A_i = P_i ⊕ SK ⊕ OTP, MOB_NO, e-mail, MAX_TRUST, MIN_TRUST and CUR_TRUST in its database. The Web server S can assign random trust values to different users according to its trust management policies. The Web server S can decide the fixed MAX_TRUST value that represents the maximum trust, fixed MIN_TRUST value that represents the minimum trust and variable CUR_TRUST value that represents the current trust value assigned to the user U_i. The Web server S will assign lower MAX_TRUST values for the users who opt for higher security (e.g., MAX_TRUST = 10), some higher value of MAX_TRUST for intermediate security (e.g., MAX_TRUST = 50) and higher value of MAX_TRUST for low security (e.g., MAX_TRUST = 1000).

Initially, the Web server S sets CUR_TRUST value equal to MIN_TRUST value. Suppose the Web server S decides MIN_TRUST to be 0, MAX_TRUST to be 50 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 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. The CUR_TRUST value may become less than MIN_TRUST value, which happen after the login failure when CUR_TRUST = MIN_TRUST.

The Web server S chooses a random value N_s and computes CK = H(N_s | URL | PK). The Web server S chooses the value of N_s in such a way that the value of CK must be unique for each user. 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 on the user’s machine when the user U_i authenticates itself successfully to the Web server S.

Step 2: S → U_i: CK.

The Web server S sends SMS of successful or failed login attempt to the user U_i on the registered MOB_NO so that user U_i will be notified of any attempt made by the attacker to login using his identity ID_i and password P_i.
3.2. Login Phase

Case 1: Client Machine Contains Cookie

User U_i establishes a connection with the Web server S using the SSL protocol. In the SSL protocol, the Web server S authenticates itself to user U_i with its public key certificate. Then 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 and sends it to the Web server S. 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 such as the Internet without using SSL protocol. The user U_i submits his identity ID_i and password P_i to the Web browser. If the user U_i’s machine contains cookie CK, then user U_i’s Web browser computes dynamic identity and password verifier information K_i = H(ID_i | URL | PK | P_i | SS | CK) and submits K_i and CK to the Web server S as shown in Figures 1 and 2.

Case 2: Client Machine Does Not Contain Cookie

User U_i agrees on SSL session key SS with the Web server S using the SSL protocol as shown in Case 1. Then all the subsequent messages of this protocol are transmitted in the open without using SSL protocol. User U_i submits his identity ID_i and password P_i to the Web browser. If user U_i’s machine does not contain cookie CK, then the Web browser chooses random nonce (any random value) value N_r, computes B_i = N_r ⊕ H(P_i), C_i = ID_i ⊕ SS and D_i = H(ID_i | SS | P_i | N_r). The Web browser of user U_i submits B_i, C_i and D_i to the Web server S as shown in Figure 3.

3.3. Authentication Phase

Case 1: Client Machine Contains Cookie

The Web server S recognizes 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 OTP as OTP = T_i ⊕ H(SK) because the Web server S knows its private key SK. Then the Web server S computes P_i as P_i = A_i ⊕ SK ⊕ OTP and computes the dynamic identity and password verifier information K_i' = H(ID_i | URL | PK | P_i | SS | 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.

Option 1

If CUR_TRUST value is more than or equal to MIN_TRUST value then the web server S chooses a
random value of $N_k$, computes $M_i = N_k \oplus H(ID_i | SS | P_i)$, $Q_i = H(ID_i | N_k | P_i | SS)$ and sends $M_i$ and $Q_i$ to the web browser of user $U_i$. The Web browser of user $U_i$ computes $N_i = M_i \oplus H(ID_i | SS | P_i)$, and $Q_i'$ with the received value of $Q_i$ to validate that the messages are sent by the legitimate Web server $S$ and not tampered during transmission. This equivalency

S. K. Sood
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 user \( U_i \) and Web server \( S \) is achieved as shown in Figure 1. Afterwards, the web server \( S \) checks \( \text{CUR\_TRUST} \) value in its database corresponding to the user identity \( \text{ID}_i \). If the \( \text{CUR\_TRUST} \) value stored in the database of Web server \( S \) is less than \( \text{MAX\_TRUST} \) value, then the \( \text{CUR\_TRUST} \) value is incremented by one (\( \text{CUR\_TRUST} = \text{CUR\_TRUST} + 1 \)) after successful login attempt by user \( U_i \) on the Web server \( S \). Finally, user \( U_i \) and Web server \( S \) agree on the common session key as \( S_k = H(\text{SS} | \text{ID}_i | N_d | \text{CK} | \text{ID}_i) \). Afterwards, all the subsequent messages between user \( U_i \) and Web server \( S \) are XOR'ed with the session key. Therefore, either user \( U_i \) or Web server \( S \) can retrieve the original message because both of them know the common session key. If user \( U_i \) fails to authenticate itself to the Web server \( S \), then Web server \( S \) decreases the \( \text{CUR\_TRUST} \) value by one (\( \text{CUR\_TRUST} = \text{CUR\_TRUST} - 1 \)) and the server \( S \) removes the cookie \( \text{CK} \) from the user's machine.

Option 2

If \( \text{CUR\_TRUST} \) value is less than \( \text{MIN\_TRUST} \) value, then Web server \( S \) computes \( N_d = \text{MIN\_TRUST} - \text{CUR\_TRUST} \) and chooses random \( \text{TRUST\_BITS} \) value having bits equal to the value of \( N_d \). Suppose the value of \( N_d \) is 3. Then the number of bits in \( \text{TRUST\_BITS} \) value will be 3. Then Web server \( S \) computes \( Z_i = N_d \oplus \text{ID}_i \oplus \text{SS} \oplus H(\text{Pi}) \), \( R_i = N_d \oplus \text{TRUST\_BITS} \), \( V_i = H(\text{ID}_i | N_d | \text{Pi} | \text{SS} | \text{TRUST\_BITS}) \) and sends \( Z_i, R_i \) and \( V_i \) to the Web browser of user \( U_i \). The Web browser of legitimate user \( U_i \) can compute the value of \( N_d \) as \( N_d = Z_i \oplus \text{ID}_i \oplus \text{SS} \oplus H(\text{Pi}) \), \( \text{TRUST\_BITS} \) as \( \text{TRUST\_BITS} = R_i \oplus N_d \) and \( V'_i = H(\text{ID}_i | N_d | \text{Pi} | \text{SS} | \text{TRUST\_BITS}) \) and verifies the computed value of \( V'_i \) with the received value of \( V_i \). Hence, the mutual authentication between user \( U_i \) and Web server \( S \) is achieved as shown in Figure 2. Finally after successful authentication, user \( U_i \) and Web server \( S \) agree on the common session key as \( S_k = H(\text{SS} | \text{ID}_i | N_d | \text{CK} | \text{ID}_i) \). Afterwards, all the subsequent messages between user \( U_i \) and Web server \( S \) are XOR'ed with the session key. Therefore, either user \( U_i \) or Web server \( S \) can retrieve the original message because both of them know the common session key. Then Web server \( S \) resets the \( \text{CUR\_TRUST} \) value equal to \( \text{MIN\_TRUST} \) value after successful authentication. If user \( U_i \) fails to authenticate itself to Web server \( S \), then Web server \( S \) decreases the \( \text{CUR\_TRUST} \) value by one (\( \text{CUR\_TRUST} = \text{CUR\_TRUST} - 1 \)) and the server \( S \) removes the cookie \( \text{CK} \) from the user's machine. On the other hand, the attacker has to guess the values of \( \text{SS}, \text{ID}_i, N_d, \text{TRUST\_BITS} \) and \( \text{Pi} \) to compute the common session key as \( S_k = H(\text{SS} | \text{ID}_i | N_d | \text{CK} | \text{TRUST\_BITS} | \text{Pi}) \). The computational efforts required by the attacker to find the \( \text{TRUST\_BITS} \) value increases exponentially with each login failure because the number of bits in \( \text{TRUST\_BITS} \) increases by one after each login failure as shown in Figure 4.

Case 2: Client Machine Does Not Contain Cookie

Web server \( S \) computes \( \text{ID}_i \) from \( C_i \) as \( \text{ID}_i = C_i \oplus \text{SS} \) and recognizes user \( U_i \) from its identity \( \text{ID}_i \). After that, Web server \( S \) computes OTP as \( \text{OTP} = T_i \oplus H(\text{SK}) \) because Web server \( S \) knows its private key \( \text{SK} \). Then, Web server \( S \) computes \( \text{Pi} \) as \( \text{Pi} = A_i \oplus \text{SK} \oplus \text{OTP} \) and \( \text{Ni} \) from \( \text{Bi} \) as \( \text{Ni} = B_i \oplus H(\text{Pi}) \). Afterwards, Web server \( S \) computes \( D_i = H(\text{ID}_i | \text{SS} | \text{Pi} | N_i) \) and verifies it with the received value of \( D_i \). If both values are equal, Web server \( S \) proceeds to the next step. Otherwise, the login request from user \( U_i \) is rejected. Web server \( S \) chooses random nonce value \( \text{Ni} \) and computes \( E_i = \text{Ni} \oplus H(\text{Pi}) \), \( N_d = |\text{MIN\_TRUST} - \text{CUR\_TRUST}| \) and chooses random \( \text{TRUST\_BITS} \) value having bits equal to the value of \( N_d \), where \( |\text{MIN\_TRUST} - \text{CUR\_TRUST}| \) represents modulus or positive value of the difference between \( \text{MIN\_TRUST} \) and \( \text{CUR\_TRUST} \). Then Web server \( S \) computes \( Z_i = N_d \oplus \text{ID}_i \oplus \text{SS} \oplus H(\text{Pi}) \), \( R_i = N_d \oplus \text{TRUST\_BITS} \), \( F_i = H(\text{Ni} | \text{N}_r | \text{ID}_i | N_d | \text{Pi} | \text{SS} | \text{TRUST\_BITS}) \) and sends \( E_i, Z_i, R_i \) and \( F_i \) to the Web browser of user \( U_i \). The Web browser can compute \( N_i \) from \( E_i \) as \( N_i = E_i \oplus H(\text{Pi}) \), value of \( N_d \) as \( N_d = Z_i \oplus \text{ID}_i \oplus \text{SS} \oplus H(\text{Pi}) \), \( \text{TRUST\_BITS} \) as \( \text{TRUST\_BITS} = R_i \oplus N_d \) and \( F'_i = H(\text{Ni} | \text{N}_r | \text{ID}_i | N_d | \text{Pi} | \text{SS} | \text{TRUST\_BITS}) \) 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. Hence, the mutual authentication between user \( U_i \) and Web server \( S \) is achieved as shown in Figure 3. Afterwards, Web server \( S \) checks \( \text{CUR\_TRUST} \) value in its database corresponding to the user identity \( \text{ID}_i \). If \( \text{CUR\_TRUST} \)
value stored in the database is more than or equal to MIN_TRUST value but less than MAX_TRUST value, Web server S increases the CUR_TRUST value by one (CUR_TRUST = CUR_TRUST + 1) after successful authentication. If CUR_TRUST value stored in the database is less than MIN_TRUST, Web server resets the CUR_TRUST value equal to MIN_TRUST value after successful authentication. After successful authentication, Web server S stores the cookie CK on the user’s machine. Then user Uᵢ and Web server S agree on the common session key as $S_k = H(SS | ID_i | N_d | N_r | P_i | N_i | TRUST\_BITS)$. Afterwards, all the subsequent messages between user Uᵢ and Web server S are XORed with the session key. Therefore, either user Uᵢ or Web server S can retrieve the original message because both of them know the common session key. If user Uᵢ fails to authenticate itself to Web server S, Web server S decreases the CUR_TRUST value by one (CUR_TRUST = CUR_TRUST - 1).

The computation efforts required from the user to log in to the Web server is least if the user’s machine contains cookies and the CUR_TRUST value stored on the Web server is equal to or more than MIN_TRUST value corresponding to that user, as shown in Figure 1. However, the computation cost of login increases for the user having valid cookie stored on the user’s machine, but the CUR_TRUST value stored on the Web server is less than MIN_TRUST value corresponding to that user, as shown in Figure 2.

The computation efforts required from the user to log in to the Web server is much higher as compared to Figures 1 and 2 if the user’s machine does not contain valid cookies corresponding to that user. Furthermore, the cost of computation for the user to log in to the Web server increases exponentially with each login failure, as shown in Figure 3. Overall, the complete concept has been represented in Figure 4.

3.4. Password Change Protocol

The legitimate user Uᵢ authenticates itself to Web server S using the above protocol. Once the mutual authentication between user Uᵢ and Web server S is achieved, user Uᵢ submits $Y_i = SS \oplus P_i \oplus P_i^{\text{new}}$ and $X_i = H(ID_i | P_i | SS | P_i^{\text{new}})$ to Web server S. Web server S retrieves $P_i^{\text{new}}$ from $Y_i$ as $P_i^{\text{new}} = Y_i \oplus SS \oplus P_i$, computes $X_i^* = H(ID_i | P_i | SS | P_i^{\text{new}})$ 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 Uᵢ and not tampered during transmission. Afterwards, Web server S updates the value of $A_i = P_i \oplus SK \oplus OTP$ and $T_i = OTP \oplus H(SK)$ stored in its database with $A_i^{\text{new}} = P_i^{\text{new}} \oplus SK \oplus OTP^{\text{new}}$ and $T_i^{\text{new}} = OTP^{\text{new}} \oplus H(SK)$, and the password gets changed. The password change is allowed by Web server S only after verifying the request using SMS from registered mobile phone number (MOB_NO).
4. SECURITY ANALYSIS

The security of messages in online transaction inside communication channel is managed with SSL protocol. The proposed cookies-based virtual password authentication protocol uses SSL protocol to establish SSL session key (SS), and all 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 user 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 $K_i = H(\text{ID}_i \mid \text{URL} \mid \text{PK} \mid P_i \mid SS \mid CK)$ or $(B_i = N_r \oplus H(P_i), C_i = \text{ID}_i \oplus SS$ and $D_i = H(\text{ID}_i \mid SS \mid P_i \mid N_r))$ corresponding to user $U_i$, which are different for each new SSL session. With each failed login attempt, the difficulty of guessing TRUST_BITS value increases because the number of bits increases by one in TRUST_BITS value after each login failure. The guessing of TRUST_BITS values will go out of the scope of the attacker, as shown in Figures 2 and 3. The legitimate user $U_i$ can easily log in to the Web server $S$ whatever may be the TRUST_BITS and CUR_TRUST values. Therefore, the proposed protocol is secure against online dictionary attack.

2. **Offline dictionary attack:** In an 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 $(K_i = H(\text{ID}_i \mid \text{URL} \mid \text{PK} \mid P_i \mid SS \mid CK))$ or $(M_i = N_k \oplus H(\text{ID}_i \mid SS \mid P_i)$ and $Q_i = H(\text{ID}_i \mid N_k \mid P_i \mid SS))$ or $(Z_i = N_d \oplus \text{ID}_i \oplus SS \oplus H(P_i), R_i = N_d \oplus \text{TRUST_BITS}$ and $V_i = H(\text{ID}_i \mid N_d \mid P_i \mid SS \mid \text{TRUST_BITS}))$ or $(E_i = N_i \oplus H(P_i), Z_i = N_d \oplus \text{ID}_i \oplus SS \oplus H(P_i), R_i = N_d \oplus \text{TRUST_BITS}$ and $F_i = H(N_i \mid N_i \mid \text{ID}_i \mid N_d \mid P_i \mid SS \mid \text{TRUST_BITS})$. The attacker cannot 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 communication between the user and the server and then tries to find out the user’s identity ID$_i$ and password P$_i$. The user’s browser uses random nonce value N$_r$ and SSL session key SS for the generation of dynamic identity and password verifier information $K_i = H(\text{ID}_i \mid \text{URL} \mid \text{PK} \mid P_i \mid SS \mid \text{CK})$ or $(B_i = N_r \oplus H(P_i), C_i = \text{ID}_i \oplus SS$ and $D_i = H(\text{ID}_i \mid SS \mid P_i \mid N_r))$ corresponding to user $U_i$, which are different for each new SSL session. The eavesdropper cannot compute user $U_i$’s identity ID$_i$ and password P$_i$ from any of the recorded messages. Therefore, the proposed protocol is secure against an 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 cannot log in successfully in subsequent login request to the server. User $U_i$ can change his password after the user and the server authenticates each other using the protocol shown in Figures 1–3. 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 Website that is under the control of the attacker. The victim enters his valid login credentials into the fraudulent Website 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 $K_i = H(\text{ID}_i \mid \text{URL} \mid \text{PK} \mid P_i \mid SS \mid \text{CK})$ or $(B_i = N_r \oplus H(P_i), C_i = \text{ID}_i \oplus SS$ and $D_i = H(\text{ID}_i \mid SS \mid P_i \mid N_r))$ corresponding to user $U_i$, which are different for each new SSL session. The fraudulent server can ignore dynamic identity and password verifier information but cannot produce valid credentials $(M_i = N_k \oplus H(\text{ID}_i \mid SS \mid P_i)$ and $Q_i = H(\text{ID}_i \mid N_k \mid P_i \mid SS))$ or $(Z_i = N_d \oplus \text{ID}_i \oplus SS \oplus H(P_i), R_i = N_d \oplus \text{TRUST_BITS}$ and $V_i = H(\text{ID}_i \mid N_d \mid P_i \mid SS \mid \text{TRUST_BITS}))$ or $(E_i = N_i \oplus H(P_i), Z_i = N_d \oplus \text{ID}_i \oplus SS \oplus H(P_i), R_i = N_d \oplus \text{TRUST_BITS}$ and $F_i = H(N_i \mid N_d \mid \text{ID}_i \mid N_d \mid P_i \mid SS \mid \text{TRUST_BITS})$ 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 Website even when the user submits correct domain name 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 cannot impersonate as valid server because it cannot produce valid credentials \( M_i = N_k \oplus H(ID_i | SS | P_i) \) and \( Q_i = H(ID_i | N_k | P_i | SS) \) or \( Z_i = N_d \oplus ID_i \oplus SS \oplus H(P_i) \), \( R_i = N_d \oplus TRUST\_BITS \) and \( V_i = H(ID_i | N_d | P_i | SS | TRUST\_BITS) \) or \( E_i = N_r \oplus H(P_i), Z_i = N_d \oplus ID_i \oplus SS \oplus H(P_i), R_i = N_d \oplus TRUST\_BITS \) and \( F_i = H (N_i | N_r | ID_i | N_d | P_i | SS | TRUST\_BITS) \) meant for the user \( U_i \), which are unique for each new session. Therefore, the attacker cannot 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 user and the server and replays these intercepted messages. The attacker can act as the user to the server or vice versa with recorded messages. In the proposed protocol, the attacker can intercept the login request message \( \{ K_i = H(ID_i | URL | PK | P_i | SS | CK) \) and \( C_i = ID_i \oplus SS \) and \( D_i = H(ID_i | SS | P_i | N_i) \) \( \} \) corresponding to user \( U_i \), which is sent by user \( U_i \) to server \( S \). Then he starts a new session with server \( S \) by sending a login request by replaying the login request message. The attacker can authenticate itself to server \( S \) as well as to legitimate user \( U_i \) by replaying old messages but cannot compute the session key \( S_k = H(SS | P_i | N_k | CK | ID_i) \) or \( S_k = H(SS | ID_i | N_d | CK | TRUST\_BITS | P_i) \) or \( S_k = H(SS | ID_i | N_d | N_r | P_i | N_i | TRUST\_BITS) \) because the attacker does not know the value of \( ID_i, P_i, S_k, N_k, N_i, N_r, N_d \) 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 user and the server. Then the attacker tries to imitate the user to log in 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 user, as the session key \( SS \) is different for each new SSL session. Therefore, the messages cannot be replayed successfully in any other SSL session. Moreover, the attacker cannot 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 \( ID_i, T_i = OTP \oplus H(SK), A_i = P_i \oplus OTP, MOB\_NO, MAX\_TRUST, MIN\_TRUST, CUR\_TRUST \) and \( CK = H(N_s | URL | PK) \) is stolen by breaking into the server's database, the attacker does not have sufficient information to calculate user \( U_i \)'s identity \( ID_i \) and password \( P_i \), because the attacker has to guess \( SK \) and OTP correctly at the same time. It is not possible to guess \( SK \) and OTP 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 user’s password or gaining unauthorized access. Modifying or inserting messages in the proposed protocol can result in authentication failure between the user and the server but cannot allow the attacker to gain any information about the user’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 \( \{ K_i = H(ID_i | URL | PK | P_i | SS | CK) \} \) from Figure 1 or 2 protocol or \( \{ B_i = N_r \oplus H(P_i), C_i = ID_i \oplus SS \) and \( D_i = H(ID_i | SS | P_i | N_i) \} \) from Figure 3 protocol. Even after recording these messages, the attacker has to guess minimum two parameters out of \( ID_i, P_i, N_r, N_i, S_i \) and \( SS \) correctly at the same time. It is not possible to guess two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against brute force attack.

12. **Cookie theft attack**: The attacker can steal the cookie from the client’s computer using plugins or other means. Then he tries to log in to corresponding authentication server using Figure 1
or 2 protocol. Using these protocols, the attacker must know the correct identity and password corresponding to the owner of the cookie. If the attacker failed to log in to the authentication sever, the server will remove the cookie from the client’s machine. Therefore, the attacker cannot gain any meaningful information for its authentication to the server. Hence proposed protocol is secure against cookie theft attack.

5. CONCLUSION

Online dictionary attacks are one of the main concerns in password-based authentication protocols. We have specified and analyzed a cookie-based virtual password authentication protocol that is effective in thwarting online dictionary attacks because the computation cost of login to the Web server increases exponentially with each login failure for an attacker. The legitimate user can easily authenticate itself to the Web server from any machine, irrespective of whether that machine contains cookies or not. The proposed protocol is simple and fast if the user is using a valid machine. Therefore, the attacker cannot gain any meaningful information for its authentication to the server. Hence proposed protocol is secure against cookie theft attack.

REFERENCES


BIOGRAPHY

Sandeep K. Sood received his M.Tech (Computer Science & Engineering) in 1999 from the Guru Jambheshwar University Hisar (Haryana), India. He completed his Ph.D. in 2010 from Department of Electronics and Computer Engineering at Indian Institute of Technology, Roorkee (Uttarakhand), 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.