Applied Computing and Informatics (2014) **10**, 68–81

King Saud University
Applied Computing and Informatics
www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

On the security of SSL/TLS-enabled applications

Manik Lal Das *, **Navkar Samdaria**

DA-IICT, Gandhinagar, India

Received 20 August 2012; revised 31 January 2014; accepted 11 February 2014
Available online 26 February 2014

KEYWORDS

Secure Socket Layer;
Transport Layer Security;
Authentication;
Public key certificate;
Man-in-the-middle attacks;
One-time pad

Abstract SSL/TLS (Secure Socket Layer/Transport Layer Security)-enabled web applications aim to provide public key certificate based authentication, secure session key establishment, and symmetric key based traffic confidentiality. A large number of electronic commerce applications, such as stock trading, banking, shopping, and gaming rely on the security strength of the SSL/TLS protocol. In recent times, a potential threat, known as main-in-the-middle (MITM) attack, has been exploited by attackers of SSL/TLS-enabled web applications, particularly when naive users want to connect to an SSL/TLS-enabled web server. In this paper, we discuss about the MITM threat to SSL/TLS-enabled web applications. We review the existing space of solutions to counter the MITM attack on SSL/TLS-enabled applications, and then, we provide an effective solution which can resist the MITM attack on SSL/TLS-enabled applications. The proposed solution uses a soft-token based approach for user authentication on top of the SSL/TLS's security features. We show that the proposed solution is secure, efficient and user friendly in comparison to other similar approaches.

© 2014 King Saud University. Production and hosting by Elsevier B.V. All rights reserved.

* Corresponding author. Tel.: +91 79 30510617.
E-mail addresses: maniklal_das@daiict.ac.in (M.L. Das), navkar.samdaria@gmail.com (N. Samdaria).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

2210-8327 © 2014 King Saud University. Production and hosting by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.aci.2014.02.001

1. Introduction

Secure Socket Layer/Transport Layer Security (SSL/TLS) (Dierks and Rescorla, 2008) protocol is typically integrated into application for protecting data sent via HTTP between clients and servers, which is also known as HTTP over TLS (HTTPS). In an SSL/TLS-enabled application, client first sends a hello message to the server then the server, upon confirming negotiated parameters, sends the server's hello message along with server's digital certificate to the client. The server's digital certificate provides information about the server's public key, certificate validity period, owner and issuer information. Once the client authenticates the server using the server's certificate, the client and the server establish session keys, which are symmetric keys used to encrypt and decrypt information exchanged during the SSL/TLS session and to verify message integrity. As a result, the SSL/TLS-enabled application aims to provide digital certificate-based server authentication, public key based key exchange, and session key based data confidentiality using a standard symmetric key encryption (e.g., AES) algorithm. In addition, message integrity is checked using message authentication codes (Fig. 1).

The SSL/TLS protocol is composed of four subprotocols – *Handshake*, *ChangeCipherSpec*, *Record*, and *Alert* subprotocols (Dierks and Rescorla, 2008). The Handshake subprotocol allows the client to authenticate the communicating server using server's public key certificate (we note that the client authentication is an optional security support the protocol offers that can also be achieved using the client's certificate), key exchange between the client and server using public key algorithm followed by keying material generation, and data confidentiality for traffic security using symmetric key encryption algorithm. The working principle of the Handshake protocol is as follows. A client, who wants to connect

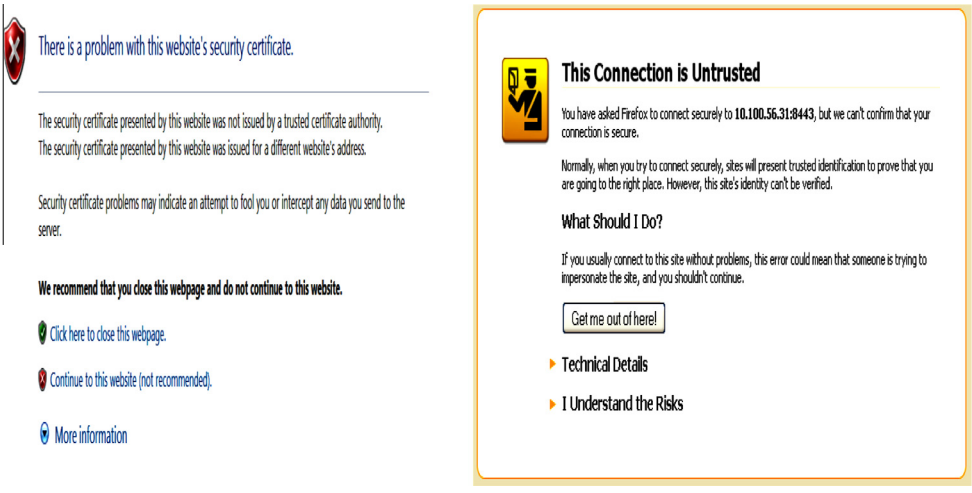

Figure. 1 Certificate warning at the client on SSL/TLS-enabled application.

to an SSL/TLS-enabled web application, is required to communicate to the server a list of ciphers and algorithms (known as cipher suites) that the client can support. Upon receiving the client's request, the server selects a suitable cipher and required algorithms from the client's cipher suites. If the server does not agree to a common cipher suite from the list given by the client, then both the client and the server are required to negotiate further toward an agreeable cipher suite. Once the server selects a suitable cipher suite from the client's list, the server sends its public key certificate along with the selected cipher suite to the client. The client now verifies the server's certificate and if the certificate is found valid then the server is authenticated. We note that the client's authentication can also be checked by the server, but it is an optional step. If the server wants to check for client's authentication then the server asks for the client's certificate. Once the server authentication is confirmed, the client and the server establish a master secret key using a key exchange algorithm based on RSA, Diffie-Hellman, or Fortezza. With this master key, both the client and the server generate key material (e.g., encryption key, MAC key) using a pseudo-random function (PRF) and other ephemeral parameters while using the SSL protocol, and for TLS protocol, it applies the HMAC and PRF to generate the key material. The Handshake subprotocol ends with the client's and server's message authentication codes of all previously exchanged messages, to protect the Handshake from message tampering. The Record subprotocol secures application data using the keys (MAC keys and encryption keys) computed in the Handshake subprotocol. Basically, the Record subprotocol is responsible for protecting the application data in the current session. The Alert subprotocol is invoked when any error or warning occurs while running the other subprotocols.

Informally, not all SSL/TLS-enabled applications require client's authentication as long as the server gets paid off for the client's service. This optional client's authentication in SSL/TLS-enabled application opens up a potential security loophole to attackers (Oppliger and Gajek, 2005) who could try to convince a naive client with a fake server certificate and if he succeeds, he would be able to capture all sensitive data from the client. This type of attack usually takes the SSL/TLS-enabled communication into some vulnerable state by tampering the security indicators and making the user believing that the connection is established with the legitimate server, but the client is actually connected to the attacker's server. For example, suppose the attacker produces a fake server certificate in place of the legitimate server certificate and the client accepts the fake certificate without checking the correctness of the certificate. After that, when the client wants to connect to the same legitimate server (for which the client has already accepted the fake certificate), the client will be routed to the attacker's server. This type of attack is known as MITM attacks on SSL/TLS-enabled application. Recently, SSL Stripping attacks (Marlinspike, 2009), (Shin and Lopes, 2011), (Zhao et al., 2012) also pose a serious threat to defeat SSL/TLS protection to web application. In SSL stripping attack, the user believes that an SSL/TLS connection has been

established with the target server, while the attacker has the ability to view the user's traffic in clear-text. The attacker suppresses the SSL/TLS peer negotiation messages and provides the user with a "stripped" version of the requested website, provoking the user to communicate to the server through an insecure channel.

1.1. Our contributions

We discuss the existing solutions to counter the MITM attacks on SSL/TLS-enabled applications. As most of the solutions consider the client authentication as an important factor, we discuss the solution space for password, graphical user interface, and token-based approaches for user authentication. We then present an efficient solution using soft-token based client's authentication that can resist SSL/TLS-enabled application from MITM attacks. We analyze the proposed solution for its security feature on top of the SSL/TLS protocol security, and provide its efficiency in comparison to other approaches.

1.2. Organization of the paper

The remainder of the paper is organized as follows. Section 2 reviews the SSL/TLS protocol, discusses MITM attacks on SSL/TLS-enabled application, and outlines existing solutions for MITM threats on SSL/TLS-enabled transactions. Section 3 presents the proposed solution for securing SSL/TLS-enabled application from MITM attacks. Section 4 analyzes the proposed solution and compares the proposed solution with related approaches. We conclude the paper in Section 5.

2. MITM Attacks on SSL/TLS-enabled Applications

An MITM attack is a form of active attack in which the attacker intercepts and selectively modifies intercepted data in order to impersonate a legitimate party involved in client and server communication. Based on intended services and business perspectives, several SSL/TLS-enabled web applications do not employ client authentication as a requirement, instead the web applications enable SSL/TLS in server authentication mode. The reason behind this is justified by saying that the server delivers services as long as the client pays the required amount to the server (e.g., online shopping facilitated by Amazon, eBay), so the client authentication is not required for such scenarios. Therefore, it is client's responsibility to check whether s/he is talking to the correct server or not. In SSL/TLS protocol, on receipt of the communicating server's certificate, the client may get a warning if any of the following occurs: (i) the Root Certifying Authority (CA) is not recognized (or trusted) by the client; (ii) the certificate is invalid/expired; and (iii) the common name (CN) of the certificate does not match with the Domain Name Server. In such cases, the client should check for a possible reason and drop the connection if not convinced with the warning message window. It is observed that

most of the time the client does not pay attention to such an important warning, and accepts the exception and stores the server's certificate in its specified location. Once the client stores a fake certificate in its memory, the damage has been done, as s/he is going to talk to a fake server though the connection is protected by SSL/TLS protocol. This type of situation opens up a door to MITM attacker (Burkhold, 2002) on SSL/TLS-enabled application. Another potential scenario is the stripping attacks (Marlinspike, 2009; Shin and Lopes, 2011; Zhao et al., 2012) on SSL-enabled web applications. SSL stripping attacks assume that an attacker is present in the middle of the victim and the server.

Many websites provide a secure connection between the client and the server by using HTTPS for protecting user credentials, confidential information such as login form, password change, money transaction and so on. Although the server response comes over HTTPS, the attacker changes all <a href = "https://..."> to <a href = "http://..."> before the victim receives the target web page. The victim believes that s/he sends the credentials over a secure connection (HTTPS) to the server, but the attacker gets the information by stripping off the connection by HTTP in place of HTTPS. It is also important for the attacker to keep a list of all HTTP substitutions that he has made so that he can send the HTTPS request back to the communicating server. Finally, the MITM attacker spoofs the client to a vulnerable state and steals useful parameters from the client. Although modern browsers support many indicators (e.g., padlock) to provide information regarding the connection, a naive client generally ignores such indications and falls into the trap. In some cases, the attacker is able to manipulate these indicators leaving no clue to the user.

In (Li and Wu, 2003), authors proposed a tamper resistant TrustBar in the browser to visualize the certificate information. In Ye and Smith, 2002, a concept of synchronized random boundaries has been proposed so as to distinguish between the authentic parts of the browser's GUI and the rendered content received by the server. Another approach that needs fewer changes at the browser is the use of dynamic security skin (Dhamija and Tygar, 2005), where the browser has a personalized area which needs to be customized by the user according to his/her will. It is believed that the attacker cannot spoof the personalized area without knowing the exact specification of the area. We now discuss some approaches for user authentication on top of the SSL/TLS protocol security strengths in order to mitigate the MITM attacks on SSL/TLS-enabled web applications.

2.1. Password-based approach

Password-based user authentication is widely used in many real-world applications. Saito et al. (2008) proposed a protocol for binding the SSL/TLS session with the client using the user's password. Saito et al's protocol binds HTTP authentication over SSL/TLS in server authentication mode. As HTTP authentication does not encrypt its channel and client does not authenticate the server, binding

the client's authentication over SSL/TLS in the server authentication mode can make the communication secure. The interesting feature of their protocol is that it binds user authentication without modifying the SSL/TLS, HTTP and HTTPS protocols. The protocol works as follows. Client and server first establish an SSL/TLS connection in the server authentication mode. After that the server authenticates the user by using HTTP authentication in arbitrary time. However, the server can control timing to obtain user's password. This is a convenient feature than the fixed timing of authentication in SSL/TLS Handshake. In order to meet these goals, Saito et al's protocol works with the following steps:

(i) Establishing SSL/TLS Handshake in server authentication mode.
  (A1) Client → Server: client_hello.
  (A2) Server → Client: server_hello, server's public key certificate (cert).
  (A3) Client → Server: $\{\text{pre-master-secret}\}_{cert}$.
  (A4) Client → Server: $\{\text{MAC}_{K_m} \text{ of (A1–A3)}\}_{K_e}$.
  (A5) Server → Client: $\{\text{MAC}_{K_m} \text{of(A1–A4)}\}_{K_e}$.
(ii) HTTP user authentication over the SSL/TLS channel.
  (B1) Client → Server: $\{\text{Request for some page}\}_{K_e}$.
  (B2) Server → Client: $\{\text{Request for user authentication}\}_{K_e}$.
(iii) Exchanging and verifying user credentials for binding authentication.
  (C1) Client → Server: $\{\text{user\_id}, h(\text{CN}, cert, h(\text{user\_id}, \text{password}, K_m))\}_{K_e}$.
  (C2) Server → Client: $\{\text{CN}, h(\text{user\_id}, \text{password}, h(\text{CN}, cert, K_m)), \text{Requested } page\}_{K_e}$.

The symbols *CN*, MAC, K_m, K_e and $h(.)$ indicate Common Name in the server certificate, Message Authentication Codes, MAC key, Encryption key and hash function, respectively. Messages A1–A5 represent the SSL/TLS handshake protocol. With the Handshake protocol, the client and server establish a master secret key by which they can derive MAC key (K_m) and encryption key (K_e). The messages B1 and B2 involve user authentication steps using HTTP and encryption key K_e. The messages C1 and C2 involve user's password. The server locally computes h(CN, *cert*, h(user_id, password, K_m)) and compares it with the one received in C1, while the client computes h(user_id, password, h(CN, *cert*, K_m)) and compares it with the one received in C2. If they validate the received content successfully, then they believe that the communication is mutually authenticated. With the step (iii), the protocol Saito et al., 2008 can bind the client's authentication using the user's password which can avoid the MITM attack on SSL/TLS session in the server authentication mode.

2.2. GUI-based approach

Authentication mechanisms such as PassFaces (PassFaces, 2009) and PASSpicture (Exum, 2007) use the natural ability of humans to recognize images. With the PASSpicture based approach normal textual password is being replaced with a

sequence of clicks on an image. There are other variants which work in similar ways (e.g., clicks on an image or draw a secret picture). Instead of using only textual password, graphical password can employ bidirectional authentication, where the server provides its authenticity to the user by displaying user's graphical password, which was chosen by the user at the time of his/her registration. Upon confirmation, the user authenticates to the server using a kind of challenge-response mechanism. Although, the GUI-based approach provides some resistance to MITM attacks, the mechanism cannot survive once the adversary captures user's graphical password.

2.3. One-Time-Password approach

There are many One-Time-Password (OTP) (Rubin, 1996) mechanisms, such as OTP manual, OTP automatic, OTP synchronous, and OTP asynchronous. These mechanisms are briefly explained as follows.

- OTP manual (e.g. Scratch card): A piece of paper or card containing OTP. The OTP must be securely printed and mailed to the customer. Using the OTP the user can login to the system and after that the OTP has no value. In other words, the adversary cannot do anything by knowing the OTP once it is used, as the user is going to use a different OTP for the next time.
- OTP automatic (e.g. SMS): Instead of scratching the OTP from a card, the OTP is sent to the registered user's mobile phone/PDA via Short Message Service (SMS). Many banking applications use this concept to protect their applications from phishing.
- OTP synchronous (e.g. Hardware/Software token): The time-synchronized OTP is same as SecurID, a hardware/software token, where each user is given a personalized token that generates an OTP every 30 or 60 s. The token's clock must be synchronized with the clock of the server.
- OTP asynchronous: In this case, the authentication server generates a random challenge and sends it to the user. The user enters the challenge into his/her token, which in turn, generates a result based on the challenge and some seed/secret value stored in the token's memory (which is known to the authentication server). Then the user sends the result back to the authentication server. Here, the challenge is valid for a short time decided by the server, and only the authentication server needs to keep track of the validity period. The advantage of the OTP asynchronous is that the clock of the user's token and the authentication server does not require to be synchronized.

We note that the OTP manual-based user authentication would make user comfortable for frequent server access. The OTP automatic is secure in comparison to the OTP manual, but a third party (e.g., mobile service provider) dependency

would be a bottleneck in this mechanism. The token-based OTP for user authentication is a secure mechanism, but synchronization is again a bottleneck.

2.4. Token-based approach

Token-based user authentication employs two-factor authentication mechanism. User enters his/her PIN and the token generates a 6-digit or 8-digit random code, which the user enters into the authentication server used for intended application. The authentication server computes the code on its own and checks whether user's code matches with its code or not. Both the user and the server should be in the same clock drift in order to accept user's token generated code. RSA SecurID (RSA SecurID, 2010) is a standard reference for token-based two-factor authentication, which has got enormous acceptance in industry and Government sectors. One can also integrate token-based user authentication to SSL–VPN enabled application for strong authentication purpose. Since the token generated code is random in nature, the MITM cannot use it for the next run by retransmitting it to the server, which will be discarded by the server. Some implementations of the SSL/TLS session-aware user authentication have been proposed in Oppliger et al. (2006, 2008) that provide the following two ways:

- Hardware-token based implementation: This approach employs impersonal tokens (that is, tokens which are not user specific) with token specific secret keys. These impersonal tokens are used for user authentication to the ACE server (RSA SecurID, 2010). The token has a small display on which the user enters his/her PIN and generates the code. The code is then used to authenticate the user. Although the approach provides security against MITM, employing such token-based implementation increases complexity and also requires extra cost.
- Soft-token based implementation: In this approach, the concept of hardware-token is converted to soft-token by emulating its functionality in software. Soft-token based solution is less expensive than the hardware-token based solution, but it suffers from additional security risks such as keylogger attacks and visual spoofing.

3. The proposed protocol

We present a soft-token based solution to mitigate the MITM threat on SSL/TLS-enabled web applications. The participating entities of our protocol are as follows:

- User/client, who wants to access SSL/TLS-enabled applications.
- Server that hosts the SSL/TLS-enabled application.

Before accessing the application, user/client requires to register with the server securely. Upon successful registration, user shares its identity (*UID*), password (*PWD*) and a pattern code (*PC*) with the server. The pattern code is an additional security measure that is added to our proposed solution. The pattern code is a shape or a sequence of matrix element $A(i,j)$ of a pattern matrix (Kumar and Raghavan, 2008). Each cell of the pattern matrix is an image that represents a character. During the registration process, the user is provided a randomly generated pattern matrix and is asked to choose some sequence of positions by typing the characters present in the pattern matrix, which then becomes the user's pattern code, *PC*. The Fig. 2 shows the user's *PC*, which is captured in the shaded region (we note that the sequence of user's *PC* is superscribed with numbers 1 to 7 in the shaded region), that is, *PC* in Fig. 2 is B{#(S7&. The protocol works as follows. When the user/client wants to access to the server, the client and server establish an SSL/TLS session key *K* using the SSL/TLS handshake protocol. After that, the server generates a soft-token containing *PC*, *UID*, and an authentication code, *AC*, where $AC = MAC(PC; h(\textit{all previous SSL/TLS messages}))$ and MAC(.) is a message authentication code. The soft-token provides user the visual character-by-character feedback of *PC* once the user inputs to *AC*. The browser will also have a small display where the compressed hash value of all the previous SSL/TLS messages will be displayed. This area can be customized by the user, thereby avoiding any visual spoofing attack. The user generates *AC* using the hash value displayed on the browser and her/his pattern code *PC*. Then, the user submits *UID* and *AC* to the server. After receiving *AC*, the server also calculates *AC* using the hash value and *PC* corresponding to the *UID*. If the computed *AC* is same as the received *AC* then the client is authenticated by the server. The user can stop inputting to *AC* if s/he does not see the pattern code corresponding to the chosen pattern at the time of registration. The process of inputting to *AC* and displaying *PC* is shown in Fig. 3. Each character of *AC* is coupled with the SSL/TLS session so that the adversary is unable to use the same message in a different SSL/TLS session. The protocol is given below:

K	t	^	(4	U	?	@
%	`	#3	&	S5	r	k
g	{2	|	~	5	76	(
B1	*	Q	/	t	u	&7
z	x	J	$	N	m	H
<	,	>	“	a	d	F
b	L	!	P	v	[	]

Figure. 2 Pattern matrix and pattern code.

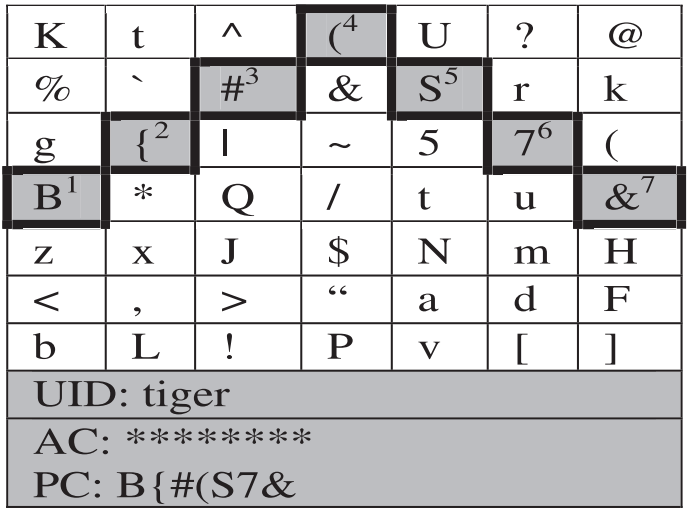

Figure. 3 Pattern matrix and authentication code.

(A1) C → S: client_hello.
(A2) S → C: server_hello, server public key certificate (cert).
(A3) C → S: $\{\text{pre-master-secret}\}_{\text{cert}}$.
(A4) C → S: MAC_{K_m} of A1–A3.
(A5) S → C: MAC_{K_m} of A1–A4.
(A6) C → S: $\{\text{Request for some information}\}_{K_e}$.
(A7) S → C: $\{\text{Request for user authentication using token}\}_{K_e}$.
(A8) C → S: $\{\text{user\_id}, \text{AC}\}_{K_e}$.
(A9) S → C: $\{\text{Requested information}\}_{K_e}$.

Messages A8 and A9 occur as many times as the server sends challenge to the client.

4. Analysis

4.1. Security analysis

The SSL (ver. 3.0)/TLS (ver. 1.0 or later) protocol provides server authentication, session key establishment and data confidentiality security services. In addition to these security services, the proposed protocol provides additional security measure for user authentication code checking using pattern matrix. The pattern matrix was selected by the user at the time of registration process and stored it securely at the server. When the user wants to connect to the server s/he has to select the appropriate patter code from the displayed pattern matrix on the legitimate server. Without knowing the pattern matrix chosen by the user, the attacker

cannot provide the correct patter matrix to the user/client, which eliminates the MITM attacks on SSL/TLS-enabled applications.

4.1.1. Server authentication

In SSL/TLS server authentication mode, the client receives the server certificate through the *server_hello* message. Upon seeing the server's certificate, the client can check its validity (by checking the issuer, subject, validity period, trust chain, purpose of the certificate) whether s/he is interacting with the correct server or not. However, the correctness checking of the server certificate by a naive user is a bottleneck. Moreover, if the client (i.e., browser) accepts (or stores somehow) a fake certificate of the (attacker) server then the user's password will get leaked to the attacker, because the user will enter his/her password after establishing the SSL/TLS connection to the attacker server in believing that s/he is talking to the correct server. Below we show that how the proposed protocol can counter this threat by authenticating the client on server stored client's information.

4.1.2. Client authentication

Many SSL/TLS-enabled applications do not employ public key certificate based client authentication. The reason behind is that the client's authentication is not required for such applications. Other possible reasons are: (i) employing public key certificate for client's authentication increases computational cost; (ii) client is required to buy certificate from a trusted authority; (iii) mobility issues for client certificate that is, if the client wants to access the server from different places across several machines or browsers, then the client has to carry its certificate in a device. Now if the server imposes client's authentication as a mandatory requirement then instead of using certificate based client authentication one can use a soft-token based approach. The proposed solution is based on soft-token for client authentication on server stored client's information, which can avoid the MITM attack on SSL/TLS-enabled applications, as explained below.

Suppose that an attacker attempts to mount an MITM attack by impersonating a user who wants to access a server enabled with the SSL/TLS protocol. We assume that the attacker establishes the SSL/TLS session with user and another SSL/TLS session with the server who hosted the SSL/TLS-enabled application. In the scenario between user and attacker, the attacker presents a false token to the user and asks for the *UID* and *AC*. Then, the attacker forwards the *UID* to the server so as to spoof the user. In this situation, the server will abort the session, as the server can identify that the *AC* given by the user does not belong to the same SSL/TLS session as the server sees a different hash of SSL/TLS messages. Furthermore, the user is not revealing her/his credentials at any stage. Therefore, the attacker cannot get user's pattern code *PC* in order to impersonate the user. The scenario is depicted in the Fig. 4 (*K1* and *K2* are SSL/TLS session keys), where the attacker will not be able to succeed in gaining anything, as two different values

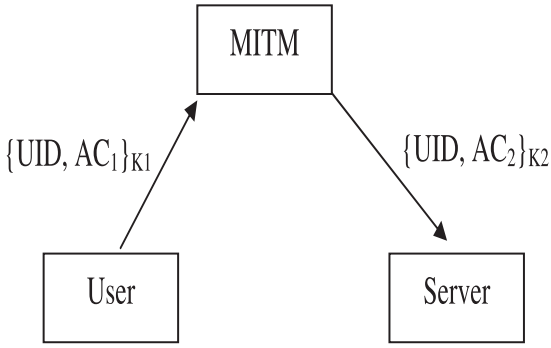

Figure. 4 Safeguard against MITM attacks on SSL/TLS-enabled application.

of *AC*s for the two different sessions are being executed by the attacker while interacting with the user and the server. Even though the attacker submits a fake server certificate to the client and if the client accepts it, the attacker cannot gain anything from the client as the attacker does not have any knowledge of the user's *AC* stored in the server. Therefore, MITM attacks cannot work in the proposed solution.

4.1.3. Replay attack

The session key of the proposed protocol is generated by pre-master secret key, client and server's random numbers, and other session specific parameters. If an adversary intercepts messages from the current or previous run of the protocol and tries to replay any messages in a new run of the protocol to get access of the server, then the adversary will not be succeeded because the server will discard the request as the messages will not pass the freshness property. The usage of random numbers of client and server in the session helps in resisting the replay attack. Therefore, the proposed protocol is not vulnerable to replay attack.

4.1.4. Password guessing attack

In the conventional password-based user authentication protocol, attacker attempts to guess user's password by intercepting messages between the user and the server. Once the attacker guesses the user's password correctly, he will then impersonate the user for gaining access to the server. In the proposed protocol, the user password is not communicated to the server. Instead, the user generates an authentication code *AC* using the patter matrix chosen by the user at the time of registration to the server. This avoids password guessing attacks in the proposed protocol. In other words, the proposed protocol prevents the password guessing attack.

Table 1 Comparison of different approaches.

Approach	Resistance to MITM attack	Computational cost	User Friendliness
Password-based	Yes	Low	Yes
GUI-based	No	High	Yes
One-Time-Password based	Yes	High	No
Hard-token based	Yes	High	No
Proposed Soft-token based	Yes	Low	Yes

4.2. Performance

We give a comparison of all approaches that we have discussed in the paper in Table 1. From the table we can see that the proposed soft-token based approach provides optimal solution in terms of security and efficiency.

The proposed solution is also usable. The user is required to input the *AC* and then s/he will be able to see the *PC* chosen at the time of registration. The approach does not require registered user to carry any device/token while s/he is roaming from one place to another place. Only thing s/he requires to remember at the time of connecting to the server is her/his patter code *PC*, which is similar to remembering a password.

5. Conclusions

We have discussed about the MITM threat to SSL/TLS-enabled web applications. We reviewed the existing approaches for user authentication and proposed a soft-token based user authentication. By enabling the proposed solution in SSL/TLS-enabled web application, the MITM attacks can be avoided. The proposed solution requires a trusted display (controlled by the legitimate server) on the browser to render the compressed value of the hash of SSL/TLS handshake messages. The proposed solution is computationally efficient and provides additional security on top of the SSL/TLS protocol's security strength.

References

Burkholder, P., 2002. SSL Man-in-the-Middle Attacks. SANS Institute Information Security.

Dhamija, R., Tygar, J.D., 2005. The battle against phishing: dynamic security skins. In: Proc. of the Symposium on Usable Privacy and Security. ACM Press, pp. 77–88.

Dierks T., Rescorla E., 2008. Transport Layer Security Protocol. Network Working Group, RFC 5246. Available from: <http://tools.ietf.org/html/rfc5246>.

Exum, T., 2007. Graphical passwords. In: Communication Security. <http://www.infosecwriters.com/textre-sources/pdf/GraphicalPasswordsTExum.pdf> (Retrieved December 2013).

Kumar, T.R., Raghavan, S.V., 2008. PassPattern System (PPS): a pattern-based user authentication scheme. In: Proc. of the International IFIP-TC6 Networking Conference on AdHoc and Sensor Networks, Wireless Networks, Next Generation Internet. ACM Press, pp. 162–169.

Li, T.Y., Wu, Y., 2003. Trust on web browser: attack vs. defense. In: Proc. of International Conference on Applied Cryptography and Network Security (ACNS), LNCS 2846, pp. 241–253.

Marlinspike, M., 2009. New Tricks for Defeating SSL in Practice. In BlackHat.

Oppliger, R., Gajek, S., 2005. Effective Protection against Phishing and Web spoofing. Communications and Multimedia Security, pp.32–41.

Oppliger, R., Hauser, R., Basin, D., 2006. SSL/TLS session-aware user authentication – or how to effectively thwart the man-in-the-middle. Comput. Commun. 29 (12), 2238–2246.

Oppliger, R., Hauser, R., Basin, D., 2008. SSL/TLS session-aware user authentication revisited. Comput. Secur. 27 (3–4), 64–70.

PassFaces, 2009. <http://www.passfaces.com/enterprise/products/web access.htm> (Retrieved June 2009).

RSA SecurID: Securing Your Future with Two-factor Authentication. <http://www.rsa.com/> (Retrieved January 2010).

Rubin, A.D., 1996. Independent one-time passwords. In: The USENIX Association of Computing Systems, vol. 9, pp.15–27.

Saito, T., Sekiguchi, K., Hatsugai, R., 2008. Authentication binding between TLS and HTTP. In: Proc. of the International Conference on Network-Based Information Systems, LNCS 5186, pp. 252–262.

Shin, D., Lopes, R. 2011. An empirical study of visual security cues to prevent the SSL stripping attack. In: Proc. of the Computer Security Applications Conference (ACSAC 2011), ACM, pp. 287–296.

Ye, Z.E., Smith., S. 2002. Trusted paths for browsers. In: Proc. of the USENIX Security Symposium, pp. 263–279.

Zhao, S., Wang, D., Zhao, S., Yang, W., Ma, C., 2012. Cookie-Proxy: a scheme to prevent SSLStrip attack. In: Proc. of the International Conference on Information and Communications Security (ICICS'12), LNCS 7618, pp. 365–372.