Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Zebin Lu

Entitled SECURE WEB APPLICATIONS AGAINST OFF-LINE PASSWORD GUESSING ATTACK: A TWO WAY PASSWORD PROTOCOL WITH CHALLENGE RESPONSE USING ARBITRARY IMAGES

For the degree of Master of Science

Is approved by the final examining committee:

Xukai Zou
Chair

Yao Liang

Feng Li

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Xukai Zou

Approved by: Shiaofen Fang 04/20/2012
Head of the Graduate Program Date

Graduate School Form 20
(Revised 9/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:

SECURE WEB APPLICATIONS AGAINST OFF-LINE PASSWORD GUESSING ATTACK: A TWO WAY PASSWORD PROTOCOL WITH CHALLENGE RESPONSE USING ARBITRARY IMAGES

For the degree of Master of Science

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Executive Memorandum No. C-22,* September 6, 1991, *Policy on Integrity in Research.**

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Zebin Lu

Printed Name and Signature of Candidate

04/20/2012

Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/c_22.html

SECURE WEB APPLICATIONS AGAINST OFF-LINE PASSWORD GUESSING ATTACK:
A TWO WAY PASSWORD PROTOCOL WITH CHALLENGE RESPONSE USING ARBITRARY IMAGES

A Thesis

Submitted to the Faculty

of

Purdue University

by

Zebin Lu

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

August 2012

Purdue University

Indianapolis, Indiana

ACKNOWLEDGEMENTS

Thanks very much to Dr. Xukai Zou, who is my research advisor for working with me, being patient with me along the research, and making precious ideas for this work. Also thanks to Dr. Yao Liang and Dr. Feng Li who have reviewed this thesis carefully and have given me many good ideas to improve the equality. Without the help of all of them, I couldn't accomplish the work.

Thanks to my parents who have continued giving me support, both materially and spiritually.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASCII	American Standard Code for Information Interchange
ATM	Automated Teller Machine
DCCP	Datagram Congestion Control Protocol
DNS	Domain Name System
DTLS	Datagram Transport Layer Security
DTPP	Dynamic Two-Way Password Protocol
EAP	Extensible Authentication Protocol
EAP-TTLS	Extensible Authentication Protocol Tunneled Transport Layer Security
FTP	File Transfer Protocol
GUI	Graphic User Interface
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
MAC	message authentication code
MITM	man in the middle
NNTP	Network News Transfer Protocol
OASIS	Organization for the Advancement of Structured Information Standards

PID	personal identification number
SID	session identifier
SMTP	Simple Mail Transfer Protocol
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TPP	Two-Way Password Protocol
TPPCA	TPP with Challenge response using Arbitrary image
Triple DES	Triple Data Encryption Algorithm
UAC	user authenticator
UDP	User Datagram Protocol
UNICODE	Unique, Universal, and Uniform Character Encoding
upu	universal password
URL	Universal Resource Locator
XMPP	Extensible Messaging and Presence Protocol

ABSTRACT

Lu, Zebin. M.S., Purdue University, August 2012. Secure Web Applications against Off-Line Password Guessing Attack: A Two Way Password Protocol with Challenge Response Using Arbitrary Images. Major Professor: Dr. Xukai Zou.

The web applications are now being used in many security oriented areas, including on-line shopping, e-commerce, which require the users to transmit sensitive information on the Internet. Therefore, to successfully authenticate each party of web applications is very important. A popular deployed technique for web authentication is the Hypertext Transfer Protocol Secure (HTTPS) protocol. However the protocol does not protect the careless users who connect to fraudulent websites from being trapped into tricks. For example, in a phishing attack, a web user who connects to an attacker may provide password to the attacker, who can use it afterwards to log in the target website and get the victim's credentials. To prevent phishing attacks, the Two-Way Password Protocol (TPP) and Dynamic Two-Way Password Protocol (DTPP) are developed. However there still exist potential security threats in those protocols. For example, an attacker who makes a fake website may obtain the hash of users' passwords, and use that information to arrange off-line password guessing attacks. Based on TPP, we incorporated challenge responses with arbitrary images to prevent the off-line password guessing attacks in our new protocol, TPP with Challenge response using Arbitrary image (TPPCA). Besides TPPCA, we developed another scheme called Rain to solve the same problem by dividing shared

secrets into several rounds of negotiations. We discussed various aspects of our protocols, the implementation and experimental results.

CHAPTER 1. INTRODUCTION

1.1 What is the World Wide Web

The World Wide Web [20], which is also known as WWW, W3 or the Web is a conceptual system which comprises of various types of interlinked documents (basically HTML, but also contains many others) available on the Internet. With the functionalities provided by a typical web browser, people can view web pages that contain a variety of contents, such as text, images, videos, which may be modified by active contents (both run on the server side and on the client side) or displayed in various styles using Cascading Style Sheet. Moreover, people can also navigate between related web pages via the hyperlinks to them.

Although the functionalities the World Wide Web provided today is much more than those in its first stage, the underlying protocol it uses to communicate the web servers and the clients is still the same, HTTP, which is further based on the network protocol suite, Transmission Control Protocol (TCP)/ Internet Protocol (IP).

Once a user asks for the resources located on a specific web server, (either by typing the URL of the web page in a web browser or by clicking a hyperlink to that page or resource, the web browser begins sending a HTTP request to the server with the Universal Resource Locator (URL) of the resource. After performing proper authentication scheme if there is any, the server then sends back to the client the

requested resource using TCP segments. Whether each TCP segment contains one or more request and response depends on the version of HTTP which is used [6 pp. 239-247]. As mentioned above, images, videos, other multimedia, active contents, or style sheet data may also be provided by the web server. Therefore, additional HTTP requests have to be made to retrieve the data. After receiving them, the web browser renders the page on the screen as specified by its HTML content using the additional data.

1.2 Popularity and Security Issues of the World Wide Web

Surfing on the Internet has already been a part in most people's lives because of its popularity and convenience. As of March 2009, the indexable web contains at least 25.21 billion pages [20]. On July 25, 2008, Google software engineers Jesse Alpert and Nissan Hajaj announced the Google Search had discovered one trillion unique URLs [19]. As of March 2012, there are over 139.0 million domains operated according to the DomainTools' announcement [14].

On the other hand, the popularity of WWW imposes a large number of underlying risks targeting not only the users but also the servers of a variety of web applications. Types of attacks include eavesdropping, spoofing, phishing attacks [10 pp. 54-55], and many others. The web applications are now used in many information sensitive areas, including on-line shopping, e-commerce, which require their users to transmit credentials on the Internet to make business activities. The result would be severe if the users couldn't protect their secrets from the adversaries on the insecure network. Therefore, to correctly authenticate a server and a user of a web application in both directions is in the predominant importance. Since the invention of the web technology including the

application layer protocol HTTP 1.0 [3], many schemes of authentication for web applications have been developed and deployed, including Basic Access Authentication, Digest Access Authentication [9], HTTPS [17] and some others.

A normal procedure deployed for authenticating a web session is to use password digest after executing the Secure Sockets Layer (SSL) protocol or the Transport Layer Security (TLS) protocol [19]. When a user asks for some resource located in a web server, the user is given back a certificate which can be used to verify the identity of the server. After executing SSL/TLS successfully both the parties share a symmetric encryption key which is used to encrypt the following data transferred in between. The user then provides a password to the server for identification check. The server checks the password with a pre-stored value. After that, the server may store a user authenticator (UAC) in the client machine to keep the user authenticated.

Using the above scheme prevents some types of network attacks, such as eavesdropping and spoofing. However, malicious people may bypass the scheme from the crack of the two parts of the protocol. For example, the malicious may produce a similar web page to the original website to trick the user to believe that the fraudulent page is the intended one. If the user fails to recognize the abnormal status, the user may provide password or other credentials to the attacker, who may use and modify that information afterwards. This problem is called a phishing attack. (There are also ways to trap users such as by sending fake emails.)

To prevent phishing attacks, researchers have been working on new schemes for many years. One of the solutions is TPP/DTPP [5], which forms the bases of our scheme, TPPCA. However there still exist potential security threats in those protocols. For

example, an attacker who makes a fake website may obtain the hash of users' passwords, and use that information to arrange off-line password guessing attacks [12 pp. 217, 241-243]. Based on TPP, we incorporated challenge responses with arbitrary images to prevent the off-line password guessing attacks in our new protocol, TPP with Challenge response using Arbitrary image (TPPCA).

Another scheme, Rain, uses shared secrets to generate challenges which accept inaccurate answers, in this way, to keep the hash of users' passwords secure from phishing attacks.

1.3 Organization of the Thesis

The rest part is arranged as the following: We introduce World Wide Web and its techniques in chapter 2. In chapter 3, we summarize various security issues regarding to web authentication, and illustrate the weakness and advantages of various existing schemes which are used to prevent different types of attacks on web applications. In chapter 4, we focus on the design and theory of one of the latest protocols, TPP. In chapter 5, we show the limitation of combining TPP with challenge responses. In chapter 6, we show how TPPCA prevents the off-line password guessing attack in addition to various other types. In chapter 7, we illustrate another possible solution, Rain scheme. In chapter 8, we summarize our implementation of TPPCA. We discuss the future work in chapter 9 and make a conclusion in chapter 10. Finally, in the Appendix we reexamine the TLS, the base protocol of TPP.

CHAPTER 2. WEB ATTACKS AND SECURITY MEASURES

2.1 Concepts of Authentication

According to Cole, E., etc. [6 p. 84], "authentication is verification that the user's claimed identity is valid, and it is usually implemented through a user password at logon time." Authentication is based on a variety of methods from users' secret passwords to people's biometric characteristics. Generally, any authentication falls into one of the following three categories:

The so-called Type 1 authentications are those that use people's knowledge of a personal secret, such as a personal identification number (PID) or a password.

The second type is based on what a user has, such as a smart card, an Automated Teller Machine (ATM) card or any other equipment.

The last type of authentications uses the characteristics of a user, which may include a fingerprint, face figure, or retina scan.

After authentication, a user is allowed to access certain computer resources and information or perform any authorized modification on those resources. Particularly, in a website scenario, users may request the resources which are located on a web server in the form of HyperText Markup Language (HTML) or any other compliant data format using HTTP protocol.

2.2 Web Authentication

Now we examine the concept of an authenticated session of web applications. As the underlying TCP protocol lacks a way to implement the authentication mechanism, HTTP itself must provide a method to authenticate users. Furthermore, HTTP or the layer above must maintain the continuity of an authenticated session up to the top business layer, which provides last-long authentication features among numbers of data transactions. As demonstrated by Gollmann, D. [10 pp. 342, 343], authenticated sessions are established on the following three layers:

Authenticated sessions exist at three conceptual layers:

The uppermost layer is business application layer, which builds up the authentication mechanism between a web application user and the corresponding service provider.

The network application layer, which lies in the middle, is the authentication layer which connects a web browser to a web server.

The bottom layer, the transport layer, provides authentication features between a TCP client and a TCP server.

Particularly, an authenticated session at the transport layer can be established with SSL/TLS on the top of TCP/IP. For the users who have a public key-private key pair and a corresponding certificate, TLS with mutual authentication can be established. However, in the real world, requiring every user possess such an identifier is never possible. Therefore web services usually use SSL/TLS with password scheme to achieve mutual authentication. An Extensible Authentication Protocol Tunneled Transport Layer Security (EAP-TTLS) model is such an example. Based on the model, the currently deployed

solution for website authentications is HTTPS protocol, which runs HTTP over TLS. The detail of HTTPS is specified in RFC 2818 [17].

For maintaining the validity of an authentication session, at the network application layer the server may create a session identifier (SID) and transmit it to its client. The client passes the SID in subsequent requests to the server. Requests contain the same SID are automatically checked and bound to the same transaction fluid which maintains the same authentication status.

Cookie is an often used in web authentication sessions to store session information in clients. A cookie is sent by a web server in a HTTP response. After that, the corresponding browser stores the cookie in a specific file and includes it in the requests of the same domain.

2.3 HTTPS and EAP-TTLS

According to RFC 2818, TLS is used as a wrapper of HTTP data, which is similar to use HTTP on the top of TCP.

To illustrate, when a web browser sends a HTTP request to a web server, if there's no pre-exist HTTPS session, it has to perform a TLS handshake, which is to perform the mutual directional authentication.

After the success of the authentication, all HTTP data is wrapped as TLS application data to provide most important security features, such as data integrity and data secrecy.

HTTPS is an example of how EAP-TTLS is implemented.

According to Gollmann, D. [10 pp. 314-316], "the Extensible Authentication Protocol (EAP) defines authentication protocols at the level of abstract message flows called methods." The methods can be built upon any possible underlying schemes.

EAP-TTLS is intended to authenticate both parties of a connection when a user connects to a server from a client machine. For example, in the scenario of a web service, a user uses a web browser to connect and request resources from a web server. The server has a certificate, which can be used by the web browser to verify the identity of the server with a public key it provides. The client uses TLS to authenticate the server through a handshake phrase and then establish a secure tunnel to the server. The user is authenticated by the server using a password scheme. As a result, EAP-TTLS prevents eavesdropping and man-in-the-middle attacks in the case that the TLS tunnel has been established correctly with the intended server, such as an intended web server or website.

2.4 Pitfall of EAP-TTLS

According to Gollmann, D. [10 p. 344], in the EAP-TTLS scenario, including the use of HTTPS, the authentication session is safe as long as the web browser, under the user's instruction, connects to the intended website. However, there exist some situations when a user tempts to make a connection to the intended server the attacker comes in the middle. For the web, this may be trigged by typing a domain name mistakenly or by clicking a fraudulent hyperlink in a phishing email. When a user is tricked into opening a TLS session with the third party, a man-in-the-middle attack becomes possible.

After the user opens a secure TLS tunnel to the attacker, the attacker can then open another TLS tunnel to the targeted server if there is a popular website with a similar

domain name. The server will ask the attacker the user's credentials, such as the user's personal identification number (PID) or password. The attacker in turn asks the user for the credentials. The user, without detecting the abnormal status, may reply the attacker with the credentials. If the attacker provides the information to the server, the server will successfully authenticate the attacker as the user. The server may afterwards create a UAC, e.g. a cookie, and send it to the attacker. From then on the attacker will impersonate the user on that website using the stored UAC. The following figure illustrates such an attack. Other than the pitfall described above, there also exists another kind of man-in-the-middle type of attack which may happen during a TLS session renegotiation phrase [18].

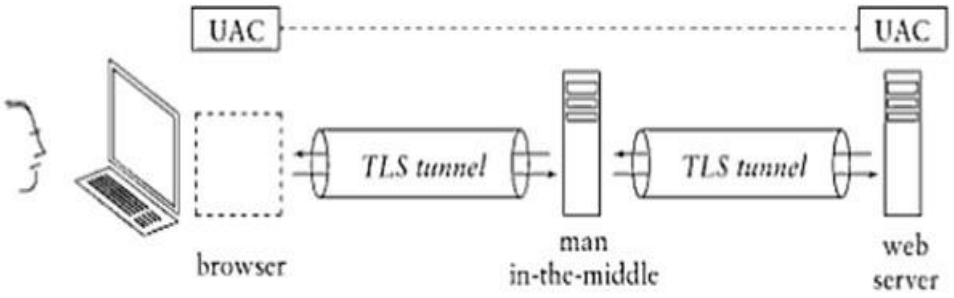

Figure 2.1 A Man-in-the-Middle Attack Breaking Application-Layer Sessions [10 p. 344]

2.5 SSL/TLS Session-aware

One of the existing methods which secure EAP-TTLS from man-in-the-middle (MITM) attacks is the SSL/TLS session-aware user authentication scheme.

As demonstrated by Oppliger, R., Hauser, R., & Basin, D. [15], "the main idea is to make the user authentication depend not only on the user's credentials, but also on state information related to the SSL/TLS session in which the credentials are being transferred

to the server." The theory behind this scheme is that the server need a way to check whether the SSL/TLS session in which the credentials is sent to the server is the same as the session in which the credentials is sent from the user. The equality of the two sessions determines the existence of a MITM attack: If the two sessions are the same, it is likely no MITM attack involved; if the two sessions are different, a MITM attack probably exists between the two parties.

In the SSL/TLS session-aware user authentication scheme, the user provides a UAC which is created using both the user's credentials and the SSL/TLS session state information. An attacker who is in the middle and holds the UAC cannot use only the credentials personate the user, because the UAC bounded to the earlier SSL/TLS session from the user to the attacker cannot be used in another session between the attacker and the targeted server. The server checks the UAC to detect anything abnormal.

However, an apparent security threat underlies SSL/TLS session-aware user authentication is that although the attacker cannot impersonate the user, without mutual authentication, the attacker can still trick the user to believe that everything goes well and to ask the user to submit the credentials. For example, by impersonating a web server of on-line shopping, an attacker may request the victim to provide credit card information to execute on-line transactions, and therefore use the credit card information in other purchases. This put the user in an unsafe environment.

2.6 Phishing Attacks and Anti-phishing Measures

As the description from the Wikipedia [16], "phishing is a way of attempting to acquire information such as usernames, passwords, and credit card details by

masquerading as a trustworthy entity in an electronic communication." Websites pretend to be from popular social networks, large commercial companies, online payment processors are commonly used to trick the careless web users. Phishing attacks are typically carried out by sending spoofed e-mail which contains a hyperlink to navigate a fake website of which the appearance and user experience are almost the same to the legitimate one.

Anti-Phishing Measures include blocked site lists, site information indicators, and some others:

In the scenario of using blocked site lists, a single central database maintains a list of fraudulent websites. The web browsers check this database before proceeding to a site. This approach is able to prevent phishing attacks if the fraudulent websites are discovered in time and the list is updated quickly. However the weakness of the scheme is that it requires universal trust in a single authority. Compromising the single authority paralyzes the whole system, and a centralized blocking list also lacks the functionality to personalize fraudulent lists according to different web users' decisions.

Site information indicator is another scheme to prevent phishing attacks. Such an indicator provides information about a website in a web browser toolbar or status bar. For example, SpoofStick [8] displays the current website's domain name in larger characters which can be examined more easily by web users. In another implementation, Firefox displays the domain name of the SSL certificate. Similarly, TrustBar [11] and a tool for Internet Explorer 7 [13] show the name of the SSL certificate authority in addition.

Another scheme, TPP, is a password protocol used in the conjunction with TLS to enable web users correctly authenticate their intended web servers, and therefore protects

the users from potential phishing attacks. We now give a detailed discussion of TPP in the following chapter.

CHAPTER 3. TPP/DTPP

In this chapter we will first discuss a major feature used by TPP/DTPP—universal password, following by the concrete design of TPP and the relevant problems. Because DTPP is with a little difference of TPP, we will briefly illustrate the uniqueness of DTPP in the end of this chapter.

3.1 Universal Password

One of the major advantages of TPP is using universal password, which is also called a master secret. With the help of the domain name, universal passwords generate unique website oriented passwords for each possible websites. This feature solves two problems together:

Psychological studies have discovered that humans can repeat with perfect accuracy about only eight meaningful items, such as digits, letters, or words [7]. If a random password is eight characters long, humans can remember only one of such a password. Thus, people tend to choose pronounceable and short passwords for easy remember, however the passwords of this category are not strong enough for off-line password guessing attacks. Even worse, many people like to choose frequently used words with numbers (such as birthday of a family member) to create passwords, which are more vulnerable to a typical dictionary attack. Controversially, to prevent attackers from using

one password of a user on one website to another, it is highly recommended to use different passwords for different websites. This produces more burdens on web users. Fortunately, TPP solves the problems by hashing the concatenation of a unique master secret and the domain name of the intended servers (the domain name can be recorded from the header of HTTP responses). As only a unique secret is required, people are able to pick up longer characters as their master secrets which are hard to be guessed by a computer.

On the other hand, using universal password introduces new security threats. One of them is that because this is the origin of the passwords for all the websites a user uses, it is devastating if the master secret is captured by an attacker. Besides, the security of a universal password relies highly on the user working environment, such as the web browsers which take the full responsibility for the user to communicate with web servers. Therefore, a malware or malicious codes, such as Trojan horse, may corrupt the system and transmit the master secret to an attacker no matter how secure the protocols seem to be. The threat occurs more often as a person uses a computer in public place without proper supervision. As a result deploying universal password scheme may produce a one point fatal weakness in web authentications.

Besides, the current use of universal password is aimed to each website separately, and no synchronization scheme is proposed. As a result, managing the master secret conveniently becomes a big issue. To clarify the idea, let’s consider the case in which a web user creates the password for each of the website being used. The user probably doesn’t set up all the accounts at the same time. From time to time, the user will be asked by different websites to change the corresponding password according to the password

aging [4 pp. 184-186] feature deployed by many websites for provide additional protection on passwords. Without proper synchronization among the password aging, the user is forced, each time a password aging event occur, to log on each website he used, and enter the new master secret to change the specific password. The user needs to record all the websites he has a password and mark the ones of which the password has been updated. Furthermore, if each of the website has a different time plan to update the corresponding user's password, the user may have to change the master secret very often, which not only requires the user's wearisome work but also keep the user busy remembering which master secret is the latest. One possible solution is to create a website service which records all the websites used by a user and automatically make updates to the passwords of those websites when the user changes the master secret by password aging event. The helpful application may also provide a method to synchronize the time of password aging events for the websites to minimize the number of times of making updates.

3.2 Design of TPP

Now we demonstrate the implementation of TPP which use universal password.

The password for any specific server/ website denoted as *pu* is computed as

$$pu = \mathrm{H}(upu, ds)$$

where the *upu* is the universal password (master secret) of the user, *ds* is the domain name of the server, and H(*upu*, *ds*) is the result of using hash function H on the concatenation of the *upu*, and the *ds*.

At the registration phrase to the website S, the user U stores in the server the user name and the hash of the password for S, which are denoted as U, and H(pu) respectively.

Each time the authentication is executed the following protocol is used:

U←→S : execute TLS and compute *ms*

U←S : *ms* <enter user id>

U→S : *ms* <U>

U←S : *ms* < H(*pu*), enter password>

U→S : *ms* <*pu*>

where the *ms* is the key generated in executing TLS in the first step. The *ms* is used to encrypt the data in the following steps of the protocol.

3.3 How does TPP Prevent Phishing Attacks

As the sophisticated phishing attacks became the one of the predominant challenges in concurrent web experience, TPP arose to solve the problem. The authors of TPP illustrate how it secures web authentications against phishing attacks as the following:

1. U ←→ M : execute TLS and compute *ms*

2. M←→S : execute TLS and compute *ms'*

3. M←S : *ms'* <enter user id>

4. U ← M : *ms* <enter user id>

5. U → M : *ms* <U>

6. M→S : *ms'* <U>

7. M←S : *ms'* < H^2(*upu*,*ds*), enter password>

8. U → M : The attack fails at this point since M cannot compute *ms*<H^2(*upu*,*dm*) > which need to be sent to U to accomplish the authentication

where *ds* is the domain name of the server and *dm* is the domain name of the attacker.

3.4 Can a DNS Break the System?

In TPP, the domain names take an important role, which determine whether the protocol execute successfully. People may ask whether TPP defeats domain name fakes.

There're several ways attacking the Domain Name System (DNS), such as manipulating the header which contains the domain information, DNS poisoning and so on. An attacker can possibly deceive a user to believe that the data packages sent by the attacker come from the intended website. He can also eavesdrop the messages of a communication. So the problem is whether the attacker can make that trick in the last step of the above protocol.

Fortunately, the fake of a website in the last step never happen because of using TLS, which uses a certificate to verify the identity of a party. It means that after executing TLS the attacker is bound to a specific domain name according to the certification, and the domain name is recorded to produce the user password on the client machine. Therefore

as long as the attacker doesn't hold the certificate of the intended website, the attacker cannot impersonate the server.

3.5 Vulnerability to a Dictionary Attack

However, theoretically TPP is not secure for off-line password guessing attacks.

To illustrate, suppose an attacker arranges a phishing attack in TPP, according to the protocol, the web server sends the attacker the hashed password for the website, denoted as H(*pu*). From then on, the attacker may start a dictionary attack to the user's master secret with all other inputs available to produce H(*pu*). This means whether the password is safe against off-line password guessing attacks is not depended on the protocol but only the quality of the password. As a result, to secure the master secret against dictionary attacks, the user must choose the master secret carefully enough. However TPP itself doesn't help its users to check the quality. People who use TPP thus have the complete responsibility to make the master secret safe. (Although some websites check the passwords entered by their users, the only thing they check is the quality of the hashes, which are produced from the master secret and therefore have a good quality. Thus, web server based password checking services don't help the users to ensure the quality of their master secrets.

CHAPTER 4. TPP WITH CHALLENGE RESPONSE

In order to make dictionary attacks infeasible, we have tried to use challenge response [4 pp. 186-190] instead of sending the hash of password in TPP. However, a typical challenge-respond scheme doesn't provide much protection against dictionary attacks. To illustrate, we now examine what happens when TPP is executed with challenge response below:

U→S: execute TLS and compute ms

U←S: ms <enter user id>

U→S: ms <U>

U→S: ms<r>

U←S: ms< eH(pu) r >

U→S: ms< pu>

In the above procedure, a user sends out a random number, denoted as *r*, which is used to generate a challenge. The server which stores the user's password could compute the response correctly. The client on behalf of the user, if accepts the response, sends the user's password back.

The reason why the above scheme is unsafe against dictionary attacks is that although the attacker who cannot obtain H(*pu*) directly, can still guess the universal password and check the guessed values by appending the intended server's domain name, hashing the result, encrypting the number *r* using it as the key, and compare with the captured value $e_{H(pu)}\,r$. Therefore, adding challenge response to TPP cannot secure low quality master secret from off-line password guessing attacks but only add a little more computational work to the attacker's computer.

What a typical challenge response scheme cannot accomplish may be added successfully by users' intervention.

CHAPTER 5. TPP WITH CHALLENGE RESPONSE USING ARBITRARY IMAGES (TPPCA)

The off-line password guessing attack target users' master secrets in TPP generally test each possible combination of frequently used characters (which include letters, numbers, and punctuations), and select the one(s) which produces the response $e_{H(pu)}\, r$ from the challenge *r*. To test the intended server the knowledge of H(*pu*), the user must provide *r* and record $e_{H(pu)}\, r$. In the other hand, these two values make it possible to perform an off-line password guessing attack. Therefore using TPP with typical challenge responses, it is never achieved to both authenticate the server successfully and prevent off-line password guessing attacks thoroughly.

We then tried to introduce the user's intervention into the authentication process. Specifically, we let the server generate an arbitrary image randomly when it receives a challenge from a web user. The generated image should contain some features which can be recognized by the user. For example, we draw some characters which contain the user name and a welcome message, then distort it in some way (such as to flip some pixels of the image randomly). The image is then encrypted using the user's password and sent back to the user. The client then decrypts the data using the password and shows the image to the user. The user examines whether the image fits the above criteria. If so, the user will tell the client to send out the password in order to complete the protocol.

5.1 Protocol of TPPCA

The complete protocol is listed below:

U→S: execute TLS and compute *ms*

U←S: *ms* <enter user id>

U→S: *ms* <U>

U←S: *ms*< $e_{H(pu)}$ Image, enter password >

U→S: *ms*<*pu*> Let the user examine the image. Send back the password or terminate the protocol.

5.2 Security Analysis

In a scenario of man-in-the-middle attack, an attacker obtains the encryption of a randomly generated image. If the attacker uses the image to arrange an off-line password guessing attack, the attacker is able to decrypt the cipher using the guessed keys but not able to verify the result images using only computer, because the image is created arbitrarily and thus there's no strict rules determining which one of the images generated by the attacker is "correct" or "wrong". Therefore there's no way to confirm the guesses of the master secret.

In another case, because the attacker doesn't know the master secret, forging such an image to conceive user is impossible. Besides, if the attacker simply transmits the cipher to the user, the client will use the attacker's domain name (instead of that of the targeted website) to decrypt the message, and probably fail to get a recognizable and reasonable

image. The user then disconnects the authentication process to prevent the attacker from obtaining the password. The following procedure describes a MITM attack in executing TPPCA:

1. U ←→ M : execute TLS and compute *ms*

2. M ←→ S : execute TLS and compute *ms'*

3. M ← S : *ms'* <enter user id>

4. U ← M : *ms* <enter user id>

5. U → M : *ms* <U>

6. M → S : *ms'* <U>

7. M ← S : *ms'* < $e_{H^2(upu,ds)}$Image, enter password> The attack fails at this point since M cannot deliver a meaningful image to the user

where *ds* is the domain name of the server and *dm* is the domain name of the phishing website/attacker.

Therefore TPPCA protects users' master secrets against man-in-the-middle attacks and off-line dictionary attacks.

5.3 Alternative Scheme

A variation of TPPCA let the user types some letters first as a challenge in a form (for example, a welcome followed by the user's name), sends them out to the server as the content of the image which the server needs to create. The server then creates such an

image using the letters, encrypts the result, and send back to the user. After decrypting the message, the user compares it with the original letters. The user authenticates the server if the image matches the letters, otherwise terminates the protocol.

The procedure is demonstrated as the below:

U→S: execute TLS and compute *ms*

U←S: *ms* <enter user id>

U→S: *ms* <U>

U->S: *ms*<*l*>

U←S: *ms*< $e_{H(pu)}$ Image, enter password >

U→S: *ms*<*pu*>

where *l* is the requirements for constructing a valid image.

5.4 Comparison of the Two Schemes

Comparing to the first scheme of TPPCA, the latter one provides an attacker more information to arrange an off-line dictionary attack as that not only the arbitrary image but also the corresponding letters are available to the attacker. This weakness gives the attacker some hints to test his guesses of master secret comparing to the first scheme. On the other hand, the second scheme of TPPCA reduces the chance of the attacker's making a recognizable image by restricting the rules to be used.

We’ve already said that the weakness of the first scheme is that by a very small chance an attacker could forge such an image with distinguishable letters. To eliminate the problem, the administrator of the website may set some constraints on the letters delivered, such as how many letters must be included and only use the popular phrases of a specific language. The user then verifies the received images with the constraints and only accepts to the qualified ones.

CHAPTER 6. RAIN SCHEME

Now we give another solution which is called *rain* scheme. We first give a general idea about the design and then show the details

6.1 General Idea

In a hashed password (message digest) scheme, a hash function provides a desirable feature to prevent trace back to the key, however it's not a good design against off-line password guessing attacks (which is also called dictionary attacks) as it provides an accurate match from a password to a hash. Suppose American Standard Code for Information Interchange (ASCII) or Unique, Universal, and Uniform Character Encoding (UNICODE) is used, there usually are only a few, if not unique, strings of human readable characters generate the same hash. Therefore once an attacker finds out such a matching string, he can be pretty sure the string of characters is the user's master secret.

In order to eliminate the problem, which is providing an accurate match of a password, we reduce the information included in each response to a challenge by giving the challenger "blurry" answers. Therefore, even if an attacker obtains an answer, without knowing enough information to test the guesses, a dictionary attack cannot succeed.

6.2 Design Detail

Here comes the rain algorithm: Each time a challenger sends to the other side a point in a two dimension space, say Q, with the coordinates (x_Q,y_Q) to identify it, and also the time he received the last massage (denoted as t_{rec}) from the other. The later one first validate the time of receiving using the constraint demonstrated in the figure 6.1 below, if it's

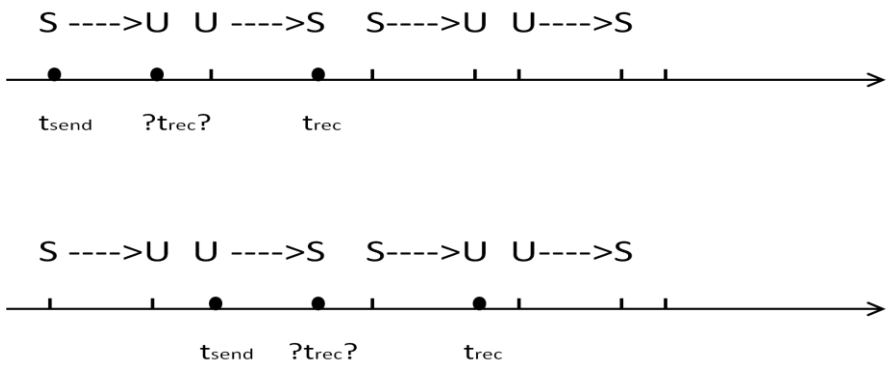

Figure 6.1 Time Validation of Rain Scheme

valid, then do the following match: The concatenation of the hashed password H(*pu*), x_Q, and y_Q is matched to an integer as the x coordinate of another point P, x_p; the concatenation of H(*pu*) and t_{rec} is matched to another integer as the y coordinate of another point P, y_p. (One way of the match is to divide a concatenation as a binary integer by another predefined large integer, and to enhance the security, matching the residue to the decimal part of π can also be used, see the figures below.)

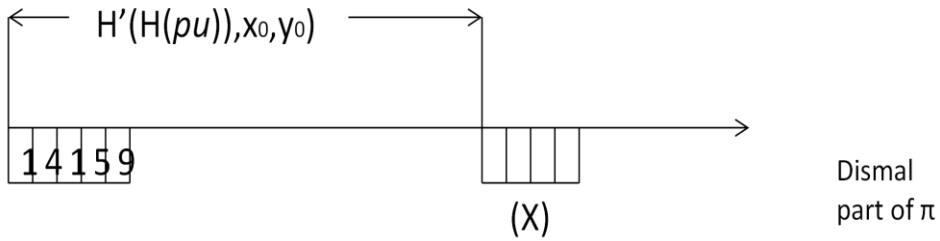

Figure 6.2 Compute X-coordinate of Point P in Rain Scheme

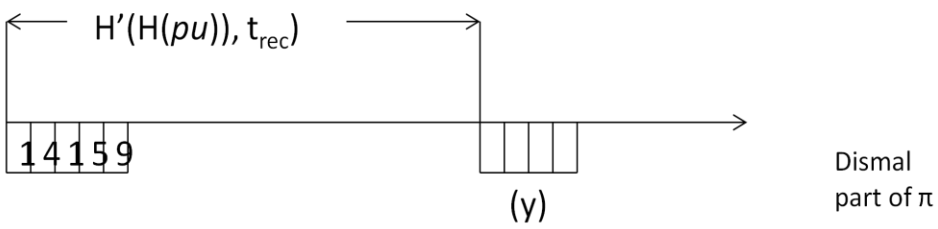

Figure 6.3 Compute Y-coordinate of Point P in Rain Scheme

Now as the responder computes the position of point P as above, instead of sending it back to the challenger directly, the responder randomly selects another point say M, denoted by its coordinates (x_M,y_M), which is within the radius of R (a predefined distance in the protocol) from P. (See figure 6.4 below) The responder then sends M back as a inaccurate but acceptable answer. The challenger, at the same time, uses the necessary knowledge to compute P, and after receiving the response, checks the answer with the above constraints using points P, M and distance R.

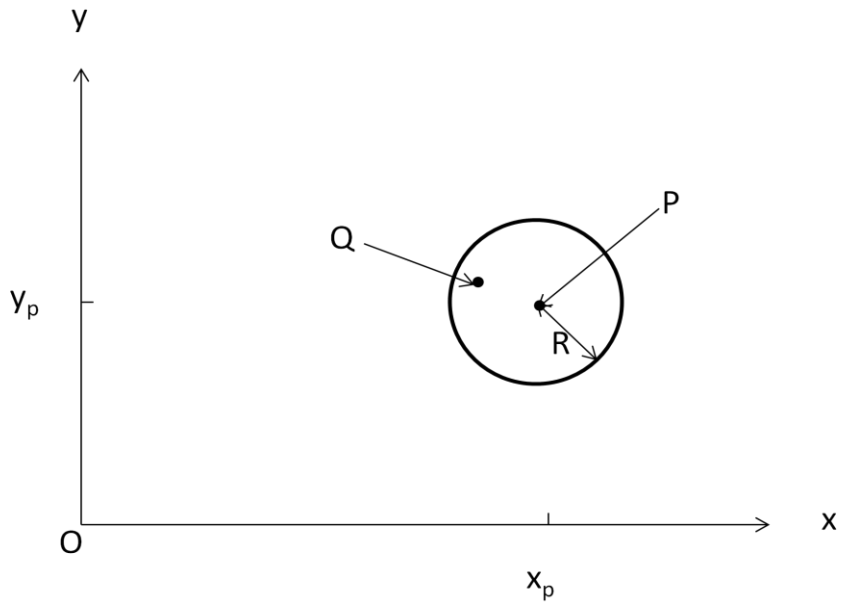

Figure 6.4 Randomly Select Q within Distance R from Point P

The procedure is summarized as below:

...

Challenger→responder: x_Q,y_Q and t_{rec}

// go ahead if t_{rec} is valid, otherwise the responder terminates the connection

Challenger←responder: x_M,y_M

// go ahead if M is within the radius of R from P, otherwise the challenger stops

…

6.3 Protocol of Rain Scheme

The rain algorithm above achieves the desired property as the required response is inaccurate but still contains part of the information of the password, H(*pu*). However, because the answers are not strictly right, an attacker in the middle of the transmission is possible to guess the point M, and by a small chance (which is determined by the parameters used in the implementation and we'll explain the detail later) he may pass the question. To prevent such a scenario, several challenge-response rounds are executed in our major protocol to minimize the possibility to any small number for an attacker to pass the authentication without knowing the password. Following is the protocol:

U←→S : execute TLS and compute *ms*

U←S : *ms* <enter user id>

U→S : *ms* <U>

U←S : *ms* < Q_0, t_{rec0} >

U→S : *ms* <Q_1, t_{rec1} >

U←S : *ms* < Q_2, t_{rec2} >

…

U←S : *ms* < Q_m, t_{recm} >

U→S : *ms* <*pu*>

where Q_i (i=1,2,…,m) is the response of the previous challenge and also serves as part of the current challenge and Q_0 is a random point to initialize the challenges. Similarly, t_{reci} is the time when the challenger receives the last message.

In the protocol, the two parties first use TLS to establish a secure channel which prevents eavesdrops and data modifications. After providing the username, the user is first challenged. The reason why not the server be challenged first is that the server is always available to everyone including the attacker. If the server first make a response to a user or an attacker (server doesn't know the identity of the requester before finish the authentication procedure), the later one will get a little information of the password after doing the first challenge (then terminate the connection each time after). After performing several connections, the attacker may obtain enough information about the password and arranges a dictionary attack. So in order to prevent this fatal weakness, in the protocol, the user is first challenged by the server; even somehow a phishing website

can do the same to the user, information collection process is much slower and more difficult than that in the previous situation because users are not always available on-line be fooled time after time by the same trick.

6.4 How to Choose the Radius

The value of R determines the probability of the success of a randomized guessing of the challenge. The probability, in turn affects how many rounds is “enough” for correctly authenticating the both parties. In another word, taking larger value of R results in higher probability of successful guesses, and thus requires more rounds of challenge-response rounds to perform the authentication. This also introduces more computational work and consumes more time to process. However, taking larger value of R also increase the number of pieces of information which is required for attacker to manage a dictionary attack. Therefore, users may be allowed to make more mistakes (such as being tricked by various fraudulent websites) while still keep their secret safe from dictionary attacks.

6.5 Other Aspects

The protocol is aimed to reduce the possibility of which a phishing attack combined with off-line password guessing attack on a web application succeeds. It provides a way to eliminate the information leaks while performs authentication correctly. However, it doesn’t prevent information leaking completely. An attacker, who personates either a subscribed user or a web server) still has the chance to pass a challenge or even achieve a piece of information of secret with no pay. Other measurements must be used together to provide a reasonable performance.

In the scenario that an attacker impersonates a legal user, if the target server allows whatever number of false guesses, the attacker will eventually collects enough information to arrange a dictionary attack off-line by keeping trying guessing the challenges. It is necessary for the server to implement one of the many time-out schemes or frozen the target account after certain number of false try. It is better to notify the user what have happened to their accounts by other means (such as making a phone call, send a mail) as well.

In another scenario, once an attacker manages a phishing attack to a subscribed user, the target user, without carefulness, may provides one piece of information of the user's secret to the attacker. Fortunately, as long as the number of mistakes below a certain threshold determined by R (which we have just discussed above), the user's secret is still safe to dictionary attack, and normally it's true because people tend to learn from making mistakes.

A more subtle problem is that attackers may communicate and change their gathered information from managing various fraudulent websites. Because people tend to use same or similar user names and passwords among various websites, the information gathered from the websites may be analyzed and combined to increase the possibility of managing dictionary attacks. In the worst case when a user uses the same user name and password among all the used websites, the total number of the pieces of the leaked information is the sum of that on each one. As we mentioned before, if too much information is unveiled to the attacker, a dictionary attack is successful, therefore users who use universal password is more likely to be targeted and the system will be unreliable.

Some suggestions to web users can alleviate the problem above if being deployed:

1. For the popular websites owned by large commercial companies and institutions, use popular search engines such as Google to open the links to the websites.

2. Always be suspect when being asked by unfamiliar websites to put into personal information such as credit card number to perform a transaction.

3. Use the tools which are embedded into web browsers to provide clearer domain information or brief description of the linked website, such as TrustBar [11].

CHAPTER 7. IMPLEMENTATION AND PERFORMANCE

7.1 Implementation

We implemented the TPPCA using Visual Basic programming language which runs in Microsoft Windows systems or any platform run .NET framework.

This procedure simulates how a client interacts with a corresponding server to authenticate each other in order to establish a secure session for the following web application

In the implementation, we suppose that before opening a connection to the server, the server and the client have already run a TLS protocol and succeeded in verifying the certificate of the server. Thus in this application the client need to first store the session key (which is created by the server), used to encrypt the following data, then computes the password for the intended website using the universal password provided by the user and the domain name of the server. (Note that we now only provide a version in the application which runs the both side of the protocol on a same machine, thus the domain name of the server is configured as 'localhost', which indicate the destination address is the local machine. A modified version of the protocol can be implemented by check up the server domain name using domain name server once the user confirms a URL.)

The first figure shows a Graphic User Interface (GUI) created by the server application and shows the status of the server.

Figure 7.1 Initial GUI of TPPCA Server

A GUI created by client program for user to enter authentication information and shows the status of the connection.

Figure 7.2 Initial GUI of TPPCA Client

After receiving a request, the server validates the user's name and sends an image encrypted using user's hash code.

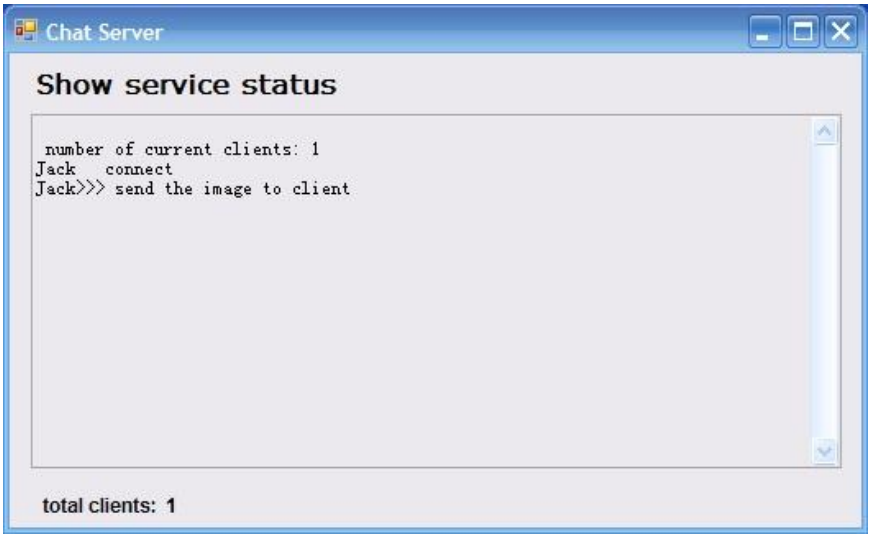

Figure 7.3 TPPCA Server Receives a Connection

The client decrypts the image and displays it in a picture box. In the implementation, we generate an arbitrary image each time which contains a welcome phrase "Hello" followed by the user's name.

Figure 7.4 TPPCA Client Decrypts the Image using the Password and Displays It

By clicking Change Image button, client asks for another image and then displays it.

Figure 7.5 User Asks for Another Image by Clicking the Change Image Button

If user click Accept button, clients sends the user's password. After validating the password, the server authenticates the user and sends a new session key for the following web application and disconnects with the client.

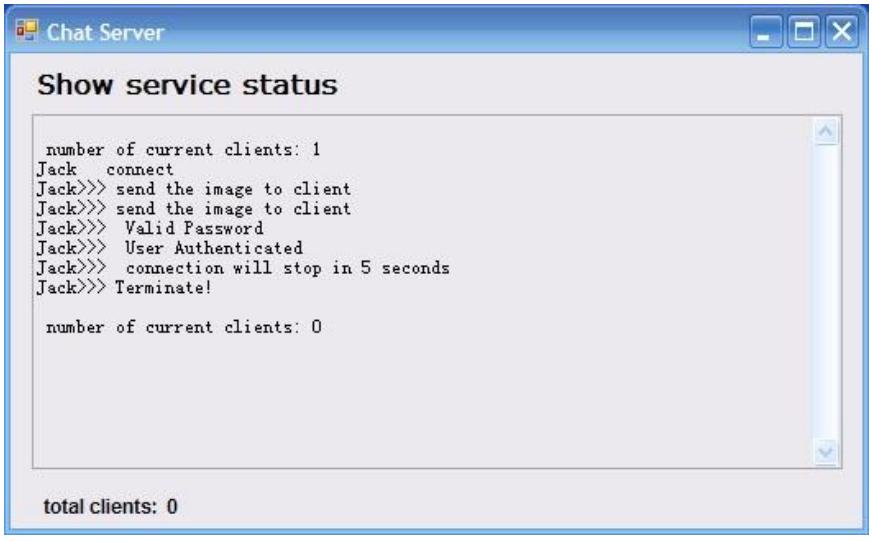

Figure 7.6 TPPCA Server Closes the Connection after Sending a New Session Key

Client gets the key which can be used in the following web application, and closes the connection.

Figure 7.7 TPPCA Client Receives the New Session Key

7.2 Performance

We now compare the performances of the TPPCA and its base protocol, TPP. The additional computation required by TPPCA includes three parts:

1. To create, encode and encrypt an image on server side:

The computer used in our implementation is a 32 bit, duo-core with 2GHz frequency and 2GB memory space. We create bitmaps which contains 200 * 200 pixels, and that takes tens of milliseconds depends on the content of the image; the encoder encodes each pixel using 24bits color scheme, and uses Triple Data Encryption Algorithm (Triple DES) for the encryption criteria, which also takes tens of milliseconds.

2. To transmit the image data:

The bitmap file created above is about 12KB. The transmission time on the network depends primarily on its bandwidth. For example, for the transmission rates are 10KB/sec,

100KB/sec and 1MB/sec, the time consumed on the transmission are 1.2sec, 0.12sec, and 12millisec respectively.

3. To decrypt and decode the image data on client side:

Like the first part, it usually takes tens of milliseconds.

The total time used above is comparatively small to a large transaction of web data.

CHAPTER 8. FUTURE WORKS

We introduced TPP with challenge response (using arbitrary image) for establishing authentication sessions for any web application. The scheme can prevent Man-in-the-middle attack, eavesdropping, phishing attack as its predecessor TPP do, as well as off-line password guessing attack which may exist in an unsecure network. However, this scheme doesn't provide any protection from off-line password guessing in user database if the server has been compromised. Because in such a case, the hash code of a user's password is available for the attacker and thus could be user directly for a password guessing attack.

Our work doesn't provide any scheme to organize a user's passwords efficiently. For example, a complete version of the scheme should provide a way to easily change all passwords which used by a user in different web application. One possible solution is to store the web application information somewhere, probably in a database on the internet. After signing in the database, user click a button to change all the passwords using the function provided by the database which automatically connects to each of the websites user uses, and change them using the new universal password user specified.

Another issue related to the above is password timeout problem. As the websites which a user uses the universal password increases, if the password for each of them has a timeout feature, user will be busy to change the universal secret frequently because it is

required by the timeout feature whenever one of the passwords times out. As another future work to do, we'll design a server-side application which allows the universal secrets generated password bypass the timeout check. The reason behind this is that the passwords generated by universal secrets are of high randomness and thus cannot be guessed by dictionary attacks. Therefore there is no need to add timeout feature to keep them secure.

For the later scheme, we'll experiment and adjust the parameters specified in the protocol to get the best performance regarding to the resources consumed (such as time and memory space) and security features (such as how many bad connections is allowed to make to a fraudulent website).

CHAPTER 9. CONCLUSION

In the thesis, we first introduced the technology of the World Wide Web and then showed how important it is in our daily lives. In the following chapters, we examined how a typical web authentication session sets up and the security problems may occur. Then we introduced the popular authentication protocols for web authentication, which include HTTPS, EAP-TTLS, SSL/TLS session-aware. The advantages and problems of each have been discussed.

After that we introduced Two Way Password Protocol (TPP) in detail, which included a feature called universal password, the procedure of the protocol and how it prevents phishing attacks. In the next chapter, we gave out a new protocol built upon TPP, which not only prevent the phishing attacks but the off-line password guessing attacks as well. As in the detail, we illustrated how TPP with a normal challenge response scheme failed to protect the secrecy from off-line password guessing attacks, and then we modified it using an arbitrary image in the protocol instead of a normal challenge and solved the problem successfully.

In the next chapter, we illustrated another scheme rain scheme, which is also a possible solution to the same problem.

We showed the details of the implementation of the first protocol and briefly discussed the efficiency. Finally, we gave a discussion other aspects among the problem and our schemes and gave a discussion of the future works.

REFERENCES

REFERENCES

[1] Alpert, J., & Hajaj, N. (2008, July 25). The official Google blog: We knew the web was big.... Retrieved from http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html

[2] An overview of the SSL or TLS handshake. Retrieved March 12, 2012 from IBM Information Center: http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/index.jsp?topic=%2Fcom.ibm.mq.doc%2Fsy10660_.htm

[3] Berners-Lee, T. J., Fielding, R. T., & Nielsen, H. F. (1996, May). Request for comments: 1945. Retrieved from http://www.ietf.org/rfc/rfc1945.txt/

[4] Bishop, M. A. (2004, November). Introduction to computer security. Boston, MA: Pearson Education

[5] Choi, T., Acharya, H. B., & Gouda, M. G. (2011, August). TPP: The two-way password protocol. Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference, 1-6. doi: 10.1109/ICCCN.2011.6005787

[6] Cole, E., Krutz, R. L., & Conley, J. W. (2005). *Network Security Bible*. Hoboken, NJ: Wiley

[7] Coombs, C., Dawes, R., & Tversky, A. (1981). *Mathematical psychology: An elementary introduction.* Ann Arbor, MI: Mathesis Press

[8] Corestreet. (2004, May). Spoofstick. Retrieved from http://www.spoofstick.com/index.html

[9] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., & Stewart, L. (1999, June). Request for comments: 2617. Retrieved from http://tools.ietf.org/html/rfc2617

[10] Gollmann, D. (2011). *Computer Security*, 3/e. Hoboken, NJ: Wiley

[11] Herzberg, A., & Gbara, A. (2006). TrustBar: Re-establishing Trust in the Web. Retrieved from http://www.cs.biu.ac.il/~herzbea/TrustBar/

[12] Kaufman, C., Perlman, R., & Speciner, M. (2002). *Network security: Private communication in a public world*, 2/e. Bergen, NJ: Prentice Hall

[13] MSDN Blogs. (2005, November 24). Better Website Identification and Extended Validation Certificates in IE7 and Other Browsers. Retrieved from http://blogs.msdn.com/b/ie/archive/2005/11/21/495507.aspx

[14] Name Intelligence. Domain counts & Internet statistics. Retrieved on March 19, 2012 from http://www.domaintools.com/internet-statistics/

[15] Oppliger, R., Hauser, R., & Basin, D. (2008, March). SSL/TLS session-aware user authentication. Computer, 41(3), 59-65. doi:10.1109/MC.2008.98

[16] Phishing. Retrieved March 3, 2012 from Wikipedia: http://en.wikipedia.org/wiki/Phishing

[17] Rescorla, E. (2000, May). Request for comments: 2818. Retrieved from http://www.ietf.org/rfc/rfc2818.txt

[18] Rescorla, E., Ray, M., Dispensa, S., & Oskov, N. (2010, February). Request for comments: 5746. Retrieved from http://tools.ietf.org/html/rfc5746

[19] Transport Layer Security. Retrieved March 6, 2012 from Wikipedia: http://en.wikipedia.org/wiki/Transport_Layer_Security

[20] The size of the World Wide Web. Worldwidewebsize.com. Retrieved from http://www.worldwidewebsize.com/

APPENDIX

APPENDIX

SSL/TLS

The following contents in this appendix use the exactly the same from [19].

TLS and its predecessor SSL are cryptographic protocols that provide communication security over the Internet. TLS and SSL encrypt the segments of network connections above the Transport Layer, using asymmetric cryptography for key exchange, symmetric encryption for privacy, and message authentication codes for message integrity.

TLS handshake in detail

The TLS protocol exchanges records, which encapsulate the data to be exchanged. Each record can be compressed, padded, appended with a message authentication code (MAC), or encrypted, all depending on the state of the connection. Each record has a content type field that specifies the record, a length field and a TLS version field.

When the connection starts, the record encapsulates another protocol – the handshake messaging protocol – which has content type 22.

A simple connection example follows, illustrating a handshake where the server is authenticated by its certificate:

Negotiation phase:

A client sends a ClientHello message specifying the highest TLS protocol version it

supports, a random number, a list of suggested CipherSuites and suggested compression methods. If the client is attempting to perform a resumed handshake, it may send a session ID.

The server responds with a ServerHello message, containing the chosen protocol version, a random number, CipherSuite and compression method from the choices offered by the client. To confirm of allow resumed hankshakes the server may send a session ID. The chosen protocol version should be the highest that both the client and server support.

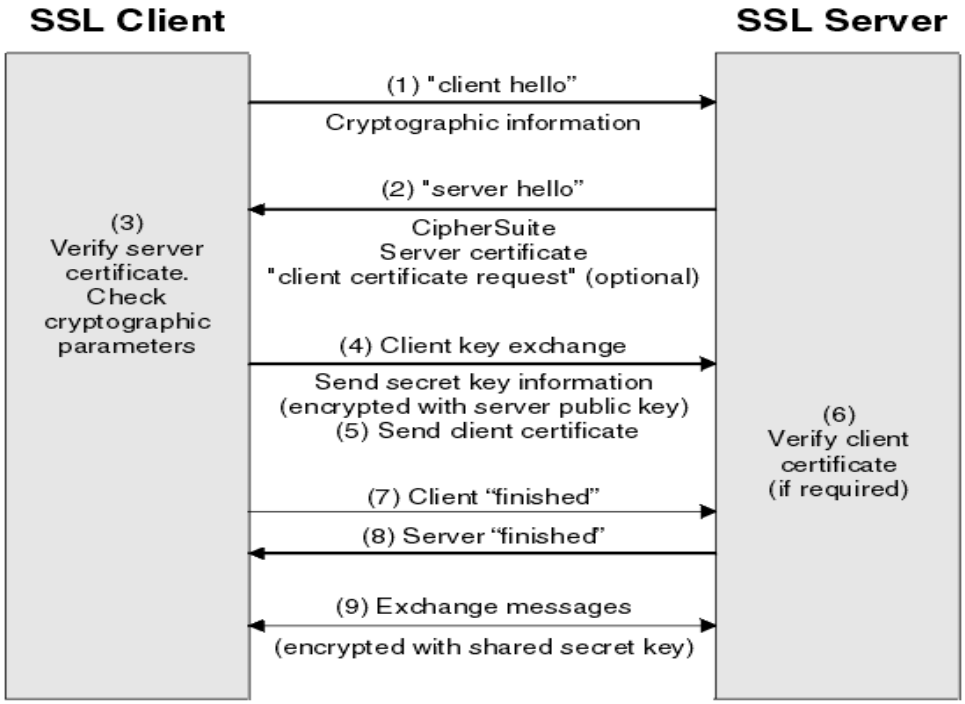

Figure A.1 SSL/TLS handshake [2]

The server sends its Certificate message.

The server sends a ServerHelloDone message, indicating it is done with handshake negotiation.

The client responds with a ClientKeyExchange message, which may contain a PreMasterSecret, public key, or nothing (This depends on the selected cipher.) this PreMasterSecret is encrypted using the public key of the serer certificate.

The client and server then user the random numbers and PreMasterSecret to compute a common secret, called the “master secret”. All other key data for this connection is derived from this master secret (and the client- and server-generated random values), which is passed through a carefully designed pseudorandom function.

The client now sends a ChangeCipherSpec record, essentially telling the server, “Everything I tell you from now on will be authenticated (and encrypted if encryption parameters were present in the server certificate).” The ChangeCipherSpec is itself a record-level protocol with content type of 20.

Finally, the client sends an authenticated and encrypted Finished message, containing a hash and MAC over the previous handshake messages.

The server will attempt to decrypt the client’s Finished message and verify the hash and MAC. If the decryption or verification fails, the handshake is considered to have failed and the connection should be torn down.

Finally, the server sends a ChangeCipherSpec, telling the client, “Everything I tell you from now on will be authenticated (and encrypted).”

The server sends its authenticated and encrypted Finished message.

The client performs the same decryption and verification.

Application phase: at this point, the “handshake” is complete and application protocol is enabled, with content type of 23. Application messages exchanged between client and server will also be authenticated and optionally encrypted exactly like in their Finished message. Otherwise, the content type will return 25 and the client will not authenticate.

TLS Applications

In applications design, TLS is usually implemented on top of any of the Transport Layer protocols, encapsulating the application-specific protocols such as HTTP, File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP), Network News Transfer Protocol (NNTP) and Extensible Messaging and Presence Protocol (XMPP). Historically it has been used primarily with reliable transport protocols such as the TCP. However, it has also been implemented with datagram-oriented transport protocols, such as the User Datagram Protocol (UDP) and the Datagram Congestion Control Protocol (DCCP), usage which has been standardized independently using the term Datagram Transport Layer Security (DTLS).

A prominent use of TLS is for securing World Wide Web traffic carried by HTTP to form HTTPS. Notable applications are electronic commerce and asset management. Increasingly, the SMTP is also protected by TLS. These applications use public key certificates to verify the identity of endpoints.