PRIVATE AND SECURE BIOMETRIC USER AUTHENTICATION IN THE WEB

By

ZAKI AIED ALSUBHI

B.S., King Abdulaziz University, SA, 2004

A thesis submitted to the Graduate Faculty of the

University of Colorado at Colorado Springs

in partial fulfillment of the

requirements for the degree of

Master of Science

Computer Science

2015
This thesis for Master of Science degree by

Zaki Aied Alsubhi

has been approved for the

Department of Computer Science

by

Terrance Boult, Computer Science, Chair

Sudhanshu Semwal, Computer Science

Jonathan Ventura, Computer Science

Date
Most of today’s secure web applications depend on Hypertext Transfer Protocol Secure (HTTPS) to provide secure communication services. Examples of these applications include e-government, online banking, and health care management, as well as a variety of other applications that manage sensitive data. HTTPS is based on top of the Secure Socket Layer (SSL) [1] or its successor the Transport Layer Security (TLS) [2]. SSL’s security services of confidentiality, integrity, and server-side authentication are widely deployed by secure web applications. However, SSL’s client-side authentication is rarely deployed.

Even when the client-side authentication of SSL/HTTPS is used, it has some limitations since it typically depends on password-based client-side authentication. In SSL/HTTPS client-side authentication, the client authenticates himself/herself using his/her private key that is usually protected by a password which can be easily stolen, given, or guessed. Also, using passwords does not provide strong non-repudiation, which is an important property of authentication methods.

The primary goal of this research is to provide a private and secure remote biometric authentication system for the web by proposing the Hypertext Transfer Protocol Biographic (HTTPB). In this protocol, the user can authenticate himself/herself to the server using his biometrics (e.g., fingerprint, irises, face). HTTPB enhances the client-side authentication and solve many limitations of HTTPS. Our proposal extends the Network Se-
curity Services (NSS) libraries’ [23] implementation of SSL/TLS in order to provide the Biocryptographic Secure Socket Layer BSSL [24]. The BSSL protocol integrates a private and secure biometric client-side authentication into the SSL/TLS.
I dedicate my dissertation work to my loving parents whose words of encouragement and push for tenacity ring in my ears. A special feeling of gratitude to my wife and my kids who helped and support me during the master period at United State of America. I also dedicate this dissertation to my brothers and many friends who have supported me throughout the process. I will always appreciate all they have done, especially Albahdal Abdullah for helping me develop my technology skills and for the many hours of proofreading.
Acknowledgments

I would like to express my deepest appreciation to all those who provided me with help and support to accomplish this journey. I am especially grateful to my advisor, Terry Boult for his valuable suggestions and guidance during my research and thesis creation. Special thanks goes to my committee members Sudhanshu Semwel and Jonathan Ventura for their consultation and guidance. My thanks also go to my friends in UCCS who studied with me some courses Yousef Alsaafari, Ahed Elmeslati, Khalid Alkhattabi and Khalid Alharbi. Last but not the least, I would like to thank everyone I forgot to mention who provided help and support along the way.
# Table of Contents

1 Introduction

1.1 Introduction to the Research Problems ............................................. 1

1.2 Contributions ..................................................................................... 4

2 Background

2.1 User Authentication ........................................................................... 6

2.1.1 Overview of User Authentication .................................................. 6

2.1.2 Properties of Strong User Authentication .................................... 7

2.1.3 Types of User Authentication ....................................................... 9

2.1.3.1 Authentication Based on Something the User Knows ............ 9

2.1.3.2 Authentication Based on Something the User Has ............. 10

2.1.3.3 Authentication Based on Something Unique (Biometrics) .... 11

2.1.3.4 Multifactor Authentication ..................................................... 16

2.2 User Authentication in the Hypertext Transfer Protocol (HTTP) ...... 18

2.2.1 Basic Authentication Scheme ...................................................... 18

2.2.2 Digest Authentication Scheme .................................................... 19

2.3 User Authentication in the Hypertext Transfer Protocol Secure(HTTPS) .. 21

2.3.1 Overview of HTTPS ................................................................. 21

2.3.2 Public Key Infrastructure (PKI) .................................................. 22

2.3.3 Secure Remote Protocol (SRP) Authentication ....................... 23

2.3.4 Pre-Shared Key (PSK) Authentication ....................................... 24

2.4 User Authentication in Web Applications ....................................... 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.1</td>
<td>Overview of User Authentication using Web Applications</td>
<td>25</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Authentication Methods in Web Applications</td>
<td>26</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Limitations of the Current Web Authentication Methods</td>
<td>28</td>
</tr>
<tr>
<td>3</td>
<td>Related Works</td>
<td>30</td>
</tr>
<tr>
<td>3.1</td>
<td>Biocryptographic Key Infrastructure (BKI)</td>
<td>30</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Biocryptographic Key Infrastructure types</td>
<td>31</td>
</tr>
<tr>
<td>3.2</td>
<td>The Biocryptographic Secure Socket Layer (BSSL)</td>
<td>34</td>
</tr>
<tr>
<td>3.2.1</td>
<td>The Biocryptographic Secure Socket Layer Types</td>
<td>34</td>
</tr>
<tr>
<td>3.2.2</td>
<td>BSSL ABCS Handshake agreement</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>The Hypertext Transfer Protocole Biocryptographic (HTTPB) Design and Implementation</td>
<td>37</td>
</tr>
<tr>
<td>4.1</td>
<td>Overview of the HTTPB’s Design and Implementation</td>
<td>37</td>
</tr>
<tr>
<td>4.2</td>
<td>Hypertext Transfer Protocol Biocryptographic Design</td>
<td>38</td>
</tr>
<tr>
<td>4.3</td>
<td>HTTPB Implementation</td>
<td>40</td>
</tr>
<tr>
<td>4.4</td>
<td>Experimental Results</td>
<td>41</td>
</tr>
<tr>
<td>5</td>
<td>Future Work and Conclusion</td>
<td>44</td>
</tr>
<tr>
<td>5.1</td>
<td>Future Work</td>
<td>44</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Operating the HTTPB in Real Life</td>
<td>45</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Learning and Experience</td>
<td>45</td>
</tr>
<tr>
<td>5.2</td>
<td>Conclusion</td>
<td>47</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>49</td>
</tr>
</tbody>
</table>
2.1 Multi-factors authenticators produce security advantages and decrease usability at the same time. ........................................ 17
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Common three authentication factors that help two parties to insure the identity of each other.</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>(a) The biological biometrics that covers unique identifiers of the person.</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>(b) The behavioral biometrics which covers any mannerisms of person.</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>Even though these photos are belong to the same person, face recognition system will not be able to match these three images successfully because of changes in pose.</td>
<td>13</td>
</tr>
<tr>
<td>2.4</td>
<td>Overview of HTTP basic authentication scheme where the user sends his password through Internet as plain text</td>
<td>19</td>
</tr>
<tr>
<td>2.5</td>
<td>Overview of HTTP digest authentication scheme, where the user’s password is hashed before send to the other party through unknown network.</td>
<td>20</td>
</tr>
<tr>
<td>2.6</td>
<td>Overview of Public Key Infrastructure (PKI). Bob applies to Registration Authority (RA) to obtain a digital certificate for his web store. The RA confirms Bob’s identity and contacts with CA to issue the certificate. When the web store visitors present the certificate to Validation Authority (VA), Bob’s identity can be confirmed.</td>
<td>24</td>
</tr>
<tr>
<td>2.7</td>
<td>In Form Based Authentication, the user sends his/her password as plain text through network and the server accepts or rejects him/her depending on the correctness of user’s password.</td>
<td>27</td>
</tr>
<tr>
<td>3.1</td>
<td>The BKI certificate format. The BKI certificate is leveraged by BSSL in order to perform biometric client-side authentication. [25]</td>
<td>32</td>
</tr>
</tbody>
</table>
3.2 Cryptographically secure biotoken overview. More detailed information can be found in [29].

3.3 Alice and Bob participate in this protocol to authenticate each other. Each of them can only obtain the other challenge after biometrically authenticating himself/herself. The authentication is performed to either.

3.4 BSSL full handshake protocol with biometric client-side authentication relying on the BKI’s ABCS. Messages labeled with asterisks (*) are optional messages [24].

4.1 Overview of the HTTPB using BSSL which relies heavily on BKI’s ABCS.

4.2 BSSL runs on top of TCP/IP like SSL/TLS layer protocol.

4.3 The average total time that required to perform the HTTPB and HTTPS protocols using the same machine.

4.4 Average password typing time versus fingerprint scanning time for enrollment and verification processes. [30]
Chapter 1

INTRODUCTION

1.1 Introduction to the Research Problems

Increased reliance on the web to perform a wide range of sensitive business activities (e.g., online banking) has caused user authentication becomes more important. There are some protocols that help the user to authenticate himself/herself through the Internet; the most popular protocol is the Hypertext Transfer Protocol Secure (HTTPS). HTTPS is introduced as an application-level protocol for distributed, collaborative, hypermedia information systems. It relies on the Secure Socket Layer (SSL) or its successor the Transport Layer Security (TLS) protocols. SSL/TLS server-side authentication with digital certificates is very common and used by many secure web applications, such as e-government, online banking, health care management, and numerous others that manage a secure data. However, the client-side authentication is rarely used, even though it is implemented in some projects such as OpenSSL and Mozilla Network Security Services (NSS). Client-side authentication is instead managed at the application layer.
The HTTPS client-side authentication has some obstacles which prevent its use, such as the challenges regarding usability, deployment, and accessibility. One of the usability challenges is that each user is required to gain a digital certificate by a Certificate Authority (CA), which commonly proves to be difficult for average web users; additionally, the lack of users who possess digital certificates prevents widespread deployment of the system. The user must also keep his/her certificate with him/her everywhere and know how to install it on his/her personal computer and keep it updated; this level of complication limits the accessibility of the system because it causes users to not install the certificate. In addition, the use of HTTPS client-side authentication has limitations in term of authenticity, usability and non-repudiation. This limitation happens because the system relies on password-based authentication. When the client authenticates himself/herself to the server, he/she uses a personal private key which is protected by a password. Password-based authentication suffers from lack of authenticity because the password can be stolen, guessed, or given away [12]. To counteract the possibility of a password’s misuse the user is also required to use long and complicated passwords and change them every period, making the passwords less usable. As a result, most users use simple passwords, write the passwords on a piece of paper, do not update passwords regularly, or use the same password across multiple systems. Thus, using password-based authentication causes additional security issues. Moreover, password-based authentication does not provide a strong non-repudiation. A user can deny his password by claiming that another person use it. Therefore, use of password-based authentication is not sufficient for protecting secure data.

There are many solutions are proposed to address the the HTTPS security and usability issues. Taylor et al. [17] proposed a solution to the HTTPS security and usability issues
by proposing a secure password-based cipher suite for TLS, which enhances the usability by not requiring a certificate-based authentication at the client side. Instead, the user is required to memories his/her password only to authenticate himself/herself. Another proposal is the Internet standard of Pre-Shared Keys (PSK) [18], which are symmetric keys, shared in advance among the communicating parties. By employing PSK, users avoid the need for public key operations. However, both solutions utilize password-based systems, and the password suffers from the security and usability limitations previously noted. Therefore, a system should not rely on passwords, especially for applications that required a high security.

The aim of this thesis is to enhance the usability, security, and non-repudiation in HTTPS client-side authentication by using privacy enhanced biometrics. We claim that using biometrics to authenticate the user will be much easier than using password because it only requires the user to submit his biometrics (i.e. fingerprint, iris, face) to authenticate himself/herself to different systems instead of having to memorize any data or to use any kind of password generator. However, privacy and security concerns regarding the user’s biometrics are the primary drawbacks encountered regarding use of biometrics in remote application authentication. Another problem is that most biometric sensors (i.e. fingerprint sensors used in smartphones or laptops) do not release biometrics signals; therefore, using these sensors to authenticate a remote user will be prohibited. To address these obstacles, we propose that, instead of using traditional biometrics, we rely on biocryptographic techniques that combine methods for handling biometric data with cryptographic methods used for security. Using these Biocryptogralic techniques addresses privacy concerns and protects the user biometrics, unlike a traditional biometrics system that uses the raw data of the
user’s biometrics. In addition, the biocryptographic technique can be leveraged to support biometric sensors that do not release biometrics signals.

The main contribution of this thesis is to provide a secure remote biometric user authentication for web applications by proposing the Hypertext Transfer Protocol Biocryptographic (HTTPB). HTTPB allows users to authenticate themselves to a web server by using their biometrics while maintaining security and privacy for the user. Our proposed HTTPB enhances the client-side authentication and addresses many of security and usability limitations of the HTTPS.

HTTPB relies on Biocryptographic Secure Socket Layer (BSSL) [24] as the HTTPS relies on SSL/TLS. BSSL proposal integrates the biometric authentication capability into the SSL/TLS’s handshake protocol. Currently, BSSL is implemented by using the OpenSSL libraries. The proposed HTTPB relies on Mozilla Firefox making use of the Network Security Services (NSS) library to provide a secure connection between two parties using mutual authentication; therefore, we implemented the BSSL into the NSS library.

1.2 Contributions

The main problem this thesis addresses is how to improve the HTTPS’s client-side authentication. This includes improving the authenticity, usability and non-repudiation of HTTPS’s client-side authentication. In order to achieve this goal, we provide the following contributions:

1. Implementing the BSSL on the Mozilla NSS library.
2. Designing the HTTPB protocol with the biocryptographic client-side authentication.
3. Extending the implementation of the Mozilla Firefox web browser to support the HTTPB using our implementation of BSSL on the Mozilla NSS library.

4. Experimenting with our implementation of HTTPB to measure the performance of HTTPB as compared to HTTPS, ensuring that the HTTPB is practical and usable.

5. Reconfiguring the Apache web server to support the HTTPB protocol.

6. Using a fingerprint scanner 1) to read legitimate users’ biometric, allowing the user to utilize a personal bioprivate key, and 2) to block non-legitimate users.
Chapter 2

BACKGROUND

2.1 User Authentication

2.1.1 Overview of User Authentication

User authentication becomes more important when we need to know the one using a specific system through the Internet is a person who has the right to take permissions or privileges. It is a fundamental building block and the first line of defense in most computer security context. It is the process of verifying the identity a person claims to, as mentioned in [4]. This process has two steps in verifying the identity of any given person. The first step is presenting an identifier to the security system, called identification. Identification occurs when a user professes an identity. The second step is generating authentication information that enhance the bond between the entity and the identifier, called verification. Verification occurs when users prove their identity. For example, users are authenticated when they provide both their user name and correct password. Identification and verification can be done by using one of several authentication types. There are three common
types of authentication which help to confirm the identity of the user: authentication based on what the user has, authentication based on what the user know, and authentication based on something unique about the user (see figure 2.1). They can be used alone or in combination [6] [17]; hence, combining any two or three types together will enhance the benefit of authentication. This approach is called multifactor authentication. The variety of authentication types helps to choose the most appropriate type for any system. All of these types can be implemented properly and used to produce a secure user authentication [5]. Every type of user authentication has advantages and disadvantages, which are discussed in more detail in subsection 2.1.3. They are rated based on certain properties, which are very important in choosing the most appropriate for the users and systems. There are three properties that help us to rate the authentication types: usability, deployability, and security. The following section discusses the user authentication types properties.

![Common three authentication factors](image)

Figure 2.1: Common three authentication factors that help two parties to insure the identity of each other.

### 2.1.2 Properties of Strong User Authentication

Every type of user authentication has properties that make it different from others. These properties are very important to know so as to determine which type of user authentication is most suitable to use for a given system. There are three important properties of
user authentication methods. These properties are usability, deployability and security, as explained below in more detail [8].

1. Usability:

The first property of the user authentication is usability, which measures the authentication methods of simplicity and ease-of-use. The authentication method must be easy to use by any users without difficult limitations. Also, the authentication method should not consume a lot of effort when being used. For example, the user does not like to remember long secrets or carry physical object everywhere he/she goes; however, he/she likes authentication method that is easy to learn without too much trouble.

2. Deployability:

The second property is the deployability of authentication method, which indicates how easy the authentication method can be deployed. Some authentication methods have requirements that make it hard to deploy. Example includes asking the user to change his client to support the authentication method or requiring the user to have a specific browser to use that authentication method. Also, the cost per user is another factor that affects the deployability of authentication method. In addition, authentication method price is one of the main factors that some organizations depend on in selecting the authentication method. Therefore, the authentication method price should be reachable and acceptable by any customer.

3. Security:

The last property, which is the most important feature of an authentication method, is a security. There are some security requirements that are necessary in authentication
method. Four of these requirements are mentioned in [8]. For example, any authentication type should be hard to spoof, meaning an attacker cannot impersonate the user. Also, having strong non-repudiation is one of the important requirements of the authentication methods, meaning the user cannot claim that somebody else participated in the authentication transaction. Another requirement is that the authentication method should be resilient to any kind of guessing such as internal observation, leaks from any verifiers, phishing or theft. Moreover, the authentication method must not rely on a third party, and the authentication cannot start without an explicit permission from the user.

2.1.3 Types of User Authentication

2.1.3.1 Authentication Based on Something the User Knows

This type of authentication is one of the most common and simplest types of user authentication. It provides the most convenient user authentication mechanism over a network. User can prove his/her identity by using something that he/she knows, such as a password, a passphrase or a Personal Identification Number (PIN). Many web applications’ authentication are based on password authentication. Health care, finance, and government organizations are examples of the fields that use a password authentication on their web application; however, this method has some problems that limit the use of it.

Unfortunately, user authentication based on what the user knows has common limitations, as mentioned in [12]. These limitations hinder the adoption of this method of user authentication. The most common limitations of password is that it can be stolen,
guessed, or given. Also, it does not provide strong non-repudiation, which is one of the most important features of strong authentication methods. Subsequently, a user can easily deny authenticating himself/herself using the password by claiming that another person has used the password. In addition, the password is not accurately known but it is memorized; therefore, it can be forgotten, in either a short or long period of time. The problem is that the user does not have only one password, but the user has different passwords for different systems. Since the user has to remember so many passwords for different uses, a user will frequently either use the same password for different systems or write passwords down. Some users tend to use easy or short passwords, which can be guessed very quickly. In addition, the user may use less than optimal passwords or use passwords that relate to him, such as his son name or his date of birth. These kinds of passwords can be vulnerable to many types of attack, such as the dictionary attack or brute force guessing. Therefore, this type of user authentication is not a good choice to use in a system that needs more careful to authenticate the users.

2.1.3.2 Authentication Based on Something the User Has

The second type of user authentication can be referred to as authentication based on something that only the user has. Authentication based on what the user has depends on a physical equipment that performs authentication [11]. It can be secure storage equipment containing passwords, such as a bankcard, smart card, ID card, or token. Another variant of what the user has is a token, which is a device that gives user a new and hard password in an easy way. It offers strong defense against brute force guessing, because it issues a very long and different password every time it is used. Therefore, it will incur less risk
of being guessed randomly. This type of authentication can be conveniently carried as the user roams. Also, Lawrence [11] states one of the advantages: when a token is used as an authenticator, and it is lost or stolen, the owner sees evidence and can therefore act accordingly. Unfortunately, it takes time to stop the other from using it. Therefore, this system has some problems as well.

The user authentication based on what the user owns has limitations in some user authentication properties. For example, roaming is restricted to locations where the appropriate token reader hardware is installed. Therefore, the user cannot use them everywhere. Another problem is that tokens are expensive in contrast with password or any soft method of authentication. Moreover, they are often lost, misplaced, or stolen, this being the biggest problem that helps intruders attack more easily. Hence, it is better to looking for more appropriate types without these limitations for future authentication.

2.1.3.3 Authentication Based on Something Unique (Biometrics)

Biometrics technology is the automated measurement of biological or behavioral traits that identify a person [10]. Biological traits include fingerprint, face, iris, palm print, ear shape and hand geometry (see figure 2.2(a)) while the behavioral characteristics include signature, keystroke dynamics and gait (see figure 2.2(b)). Biometrics provide an inextricably link between the measurement and its owner. Biometrics provide non-repudiation, as it provides proof of the transaction such that the parties who were involved cannot easily reject the transaction as unauthorized. Biometrics is not the best type since it has disadvantages, but it is very helpful when combined with other types of user authentication.
Figure 2.2: (a) The biological biometrics that covers unique identifiers of the person. (b) The behavioral biometrics which covers any mannerisms of person.

Using biometrics for authentication has significant drawbacks concerning its use in client side authentication. These drawbacks differ from one modality to others. There are five important drawbacks of the biometrics which are mentioned in [15]:

1. Noise incensed data: Biometrics data should be more accurate during registration and when using the biometric to authenticate. However, sometimes the biometric data is affected. For example, fingerprint can be affected by scarring from some negative contact while the voice can be affected by cold weather. These kinds of effects provide noisy data. Noisy data could also result from a problem that the sensor has, e.g., dirty fingerprint sensor. Noisy data leads to an increased percent of the false acceptance or false rejection rates.

2. Intra-class variation: This drawback happens when the biometric data is taken from a person during authentication is different from the data that was used to generate the template during enrollment. This drawback will affect the matching process (see
Moreover, the variations that happen by this drawback becomes more likely when the user interacts incorrectly with the sensor.

3. Distinctiveness: While it is expected that there is a big difference between different individuals biometric data, there may be a large number of people have very similar biometrics data. This limitation reduces the discriminability provided by the biometrics trait.

![Figure 2.3](image)

**Figure 2.3:** Even though these photos are belong to the same person, face recognition system will not be able to match these three images successfully because of changes in pose.

4. Nonuniversality: It is expected that every body has all kinds of biometrics data. However, it is possible for a subset of users to not have a particular traits specially for those who have disable problem. For example, if somebody does not have fingers, he/she cannot interact with biometric systems that use the fingerprint as an authentication method, or if the trait that is needed is a behavioral trait, such as a gait, and the user who needs to be authenticated does not walk, the system cannot authenticate him.

5. Spoof attack: Some of biometrics traits are very easy to mimic and use by illegitimate user. Hence, he/she enrolled as a legitimate user in order to circumvent the biometric system. This drawback will easily happen when signature or voice trait is used as identifier because there are people who can copy any signature, voice, or gait as it is and use them at the biometrics systems.
In addition, privacy is also one of concerns that can degrade the use of biometrics since some of biometrics measures do not take sufficient measures in protecting the user’s privacy, as detailed in the next subsection.

**Privacy-Enhanced Biometric**

Most biometrics measures are not respectful of the privacy of the users. For example, face and voice are obviously not a secret, and it is difficult to keep a fingerprint or iris secret from a determined attacker. Therefore, it is possible for an attacker to acquire the biometric traits of a legitimate user. Hence, the attacker can creates physical artifacts such as gummy fingers and use it as a legitimate user. Another example involves the voice trait, which can be mimicked easily, causing the biometric systems that depend on this kinds of biometrics and no other form of authentication to be attacked. In addition, some people do not like to use their biometrics traits, because of the privacy traceability. That means if the user’s biometrics data is compromised by someone who wants to follow the user, he/she can trace the victim by their biometric traits since some biometric applications are not secure enough. Moreover, the biometric authentication systems problem becomes more serious when the data associated with a biometrics feature has been compromised. Authentication systems that are based on user IDs and passwords can be changed as often as required while the user only has a limited number of biometrics features. Therefore, the user may quickly run out of biometric features that can be used for authentication if the biometrics data are compromised [9]. However, some solutions have been produced to reduce these and similar problems.
Jain et al. [10] suggest four properties that should be possessed to protect the biometrics template privacy. First, the template must be unique across databases. Second, the template should be able to be reissued easily based on the same biometric data. Third, it must be hard to obtain the original biometric template from the secure template. Fourth, the biometrics template protection scheme must not decrease the recognition of the biometrics system (False Accepting Rate and False Rejecting Rate). Jain et al. [10] classifies the literature of privacy enhanced biometrics into two types: feature transformation and biometric cryptosystems.

The feature transformation approach occurs when a transformation Function (F) is applied to the biometric template. Depending on characteristic of the transformation function, this solution can fall under one of two categories: salting and noninvertible transforms. In salting, the transformation function (F) is invertible, meaning if the key and the transformed template are compromised, the original biometric template can be retrieved. On the other hand, noninvertible transformation schemes usually apply a one-way function on the biometric template and the template cannot be inverted, even if the key is known [10].

Biometric cryptosystems can be further classified as key-binding and key generation systems, depending on how the helper data is obtained. The helper data is public information about the biometric templates. This public information is needed during matching to extract a cryptographic key from the query biometric features. Biometric cryptosystem classifications depend on how the helper data is derived. When the helper data is derived by binding a key with the biometric template, it is called a key-binding biometric cryptosystem. The key generation biometric cryptosystem is provided when the helper data is only derived from the biometric template and then the cryptographic key is generated from
the helper data and the query biometric features [10]. However, both types of template protection schemes have limitations [10]; hence, there is a need to find a way that has a perfect protection of the biometrics template, such as a combination of the biometrics with other authentication factors to respect the privacy of biometrics.

2.1.3.4 Multifactor Authentication

Multifactor authentication means combining two or more types of the user authentication. This provides an enhancement of authentication, [12]. Usually, authentication based on what the user knows and what the user has combined together to produce a stronger security. For instance, many digital tokens are combined with another factor such as an associated password, which helps to protect a lost or stolen token. One Time Password (OTP) is another example of multifactor authentication [7]. The OTP is sent to the user in near real time and must be used within a very short time. It also requires a different password in every transaction, which provides a stronger authentication type. As shown in table 2.1, when more than one type of user authentication is combined together, stronger authentication will be produced. In other words, to make the user authentication stronger and more secure, we have to combine two or more types of user authentication. Therefore, the multiple layers of authentication require an attackers to possess more or know more than if they need to spoof a single layer [4].

Since the multifactor user authentication depends on two factors or more, it has the same limitations of the types that are combined together. For example, combining passwords and token includes the limitations of password and token at the same time. This means the user has to carry the token and memorize the password for any time he/she want
<table>
<thead>
<tr>
<th>Authentication Methods Combination</th>
<th>Security Advantage</th>
<th>Usability Drawback</th>
<th>Real-World example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combining knowledge and token Authentication.</td>
<td>If the token is stolen or lost, the knowledge will protect.</td>
<td>Should memorize the knowledge and carry the token.</td>
<td>Withdrawing cash from an ATM.</td>
</tr>
<tr>
<td>Combining biometrics and ID card.</td>
<td>If one of them compromised, the other provides the security.</td>
<td>Must carry the ID and have biometrics reader.</td>
<td>Fingerprint and ID for immigration and border protection.</td>
</tr>
<tr>
<td>Combining knowledge and biometrics.</td>
<td>If one of them is compromised, the other provides security.</td>
<td>Must memorize the password and have biometrics reader.</td>
<td>Password and fingerprint to access a locker.</td>
</tr>
<tr>
<td>Combining knowledge, biometric, and ID card.</td>
<td>In case two are compromised, the third provides security.</td>
<td>Have to carry ID and biometric reader and memorize a password.</td>
<td>High security military locations requires photo ID checked by guard, plus fingerprint and password.</td>
</tr>
</tbody>
</table>

Table 2.1: Multi-factors authenticators produce security advantages and decrease usability at the same time.

to authenticate himself/herself. Table 2.1 shows some usability drawbacks of multifactor authentication [11].

Another problem with multifactors is deployability because using the protocols and technologies commonly available everywhere is important in authentication. Since there is a need for an authentication type that is widely deployed, multifactors types will not be a good choice. When there is more than one type of authentication are used to provide authentication, all of them should be available when using that type of authentication, which is not generally possible. For example, authentication schemes for the Internet at large can not rely on technology not widely deployed, such as Internet kiosk. Most web application use Hypertext Transfer Protocol, which is one of the protocols that authenticates the user
over the Internet without complicated technologies. There are two ways of HTTP authen-
tication: Basic and Digest. In the following section there is more information about HTTP
authentication.

2.2 User Authentication in the Hypertext Transfer Protocol (HTTP)

In communication between a client and server, the client communicates with a server
through a public communications network, such as the Internet, or a private communica-
tions network, such as an Intranet. HTTP is introduced as an application level protocol
for distributed, collaborative, hypermedia information system [13]. Some HTTP authenti-
cation methods are used to distribute the data between the client and server using HTTP.
The HTTP authentication framework provides several challenge response authentication
techniques. They can be used by a server to challenge a client request or by the client
to provide authentication information. The most common HTTP authentication schemes
are Basic authentication scheme and Digest authentication scheme. Both of them use a
shared secret based mechanism for access authentication [14]. Subsections 2.2.1 and 2.2.2
discusses both schemes in more details.

2.2.1 Basic Authentication Scheme

The HTTP basic authentication scheme depends on the user to authenticate him-
self/herself by using user name and password for each realm. The value of the realm must
be considered a fuzzy string, which can compare only with another realm for equality on
the same server. If the request can validate the username and password to the server, the
server will authorize the request for the protection space of the request URI [14]. The server responds with a challenge if it receives an unauthorized request for a URI within the protection space (see figure 2.4). This scheme of authentication is easy to attack since there is no security provided during authentication.

![Figure 2.4: Overview of HTTP basic authentication scheme where the user sends his password through Internet as plain text](image)

2.2.2 Digest Authentication Scheme

The HTTP digest authentication scheme does not provide a complete secure system for the World Wide Web. This scheme provides no encryption of the message content be-
tween client and server. It is based on a simple challenge response paradigm. The digest authentication scheme challenges using a nonce value. A valid response should contain a correct user name, password, given nonce value, HTTP method, and requested URI. This means the password is never sent in clear text [14]. Figure 2.5 shows how digest authentication works between the client and server. However, the HTTP digest authentication scheme is a password-based system on the server side. Hence, it has the same limitations of any password system, such as the password being forgotten or stolen. Also, it does not provide a strong non-repudiation, which is one of most important factors of user authentication.

Second, this scheme does not provide secure arrangements in advance between the server and the client to produce a secure password. The kind of password used by HTTP digest authentication is weak, therefore the digest authentication can be vulnerable by many kinds of attackers such as brute force and dictionary attackers.

![Figure 2.5: Overview of HTTP digest authentication scheme, where the user’s password is hashed before send to the other party through unknown network.](image)

The goal is to develop an authentication scheme that prevents any kind of attacker from authenticating himself/herself as a legitimate user. Therefore, Hypertext Transfer
Protocol Secure (HTTPS) is a appropriate protocol when a secure connection between the client and the server is needed.

2.3 User Authentication in the Hypertext Transfer Protocol Secure (HTTPS)

HTTP is originally used without protection of data on the Internet. However, there are two main HTTP user authentication schemes: Basic user authentication scheme and Digest user authentication scheme. These schemes do not provide a strong user authentication and depend on server-side-only authentication, which has lots of limitations. Therefore, HTTPS is introduced as an application-level protocol for distributed and collaborative a secure data remotely. HTTPS signals the web browsers to use an add encryption layer of SSL/TLS protocols to protect the privacy and integrity of exchange a secure data.

2.3.1 Overview of HTTPS

HTTPS is a layer of the Hypertext Transfer Protocol (HTTP) on top of the SSL/TLS to add the security capabilities of SSL/TLS to standard HTTP communications. SSL and TLS were designed to produce a tunnel-oriented security between the client and the server. The server side authentication is widely used by HTTPS’s applications. However, the Secure Socket Layer (SSL) or Transport Layer Security (TLS) protocol’s client side authentication is rarely used and deployed. HTTPS is a communication protocol that provides a secure communications over a computer networks with especially wide deployment on the Internet.
However, the deployment of HTTPS’s client side authentication was restrained by some obstacles. These obstacles include accessibility, usability and deployability. HTTPS protocol depends on a digital certificate, which is the main problem behind these obstacles. Users who are using HTTPS have to keep their digital certificate accessible to use HTTPS’s client side authentication. Also, users cannot access a secure web application that depend on HTTPS from just any public computer because the computer must have the digital certificate installed to gain access. One of the usability obstacles is that the user should manage his private key by himself/herself and install the digital certificate in his machine, which is not easy for the average user to do. Moreover, the user is required to have a digital certificate that is signed by trusted Certificate Authorities(CAs). This obstacle raises the deployment difficulties because not all the users can deal with trusted CA to certify their identities.

2.3.2 Public Key Infrastructure (PKI)

Public Key Infrastructure (PKI) is an infrastructure that facilitates, creates, manages, distributes, stores, and revokes digital certificates. In other words, it is a process that binds public keys with particular user identities under the Certificate Authority (CA) control. Also, PKI is a set of roles and one of these roles is asymmetric cryptography. This role requires two separate keys one of them is a private and the other is public. Although, the two of them are linked mathematically. For example, if the public key is used to verify a digital signature, the private key is used for the opposite operation which is create a digital signature. As shown in figure 2.6 the PKI attempts to provide a trusted way to solve an important problem in how to exchange these keys between two parties through
unsecured tunnel. Since there is an explosion of successful phishing and other man-in-the-middle attacks in recent years, this problem become more prevalent, and a solution more important. [19]

However, Ellison and Schneier [20] mention ten security risks of PKI, some of which are mentioned here. For example, the CA is often defined as ”trusted party” and that is not true because in cryptography a trusted authority means the user has not shared his/her secrets. Another example is the private key is saved on a conventional computer, hence it is subject to attack by viruses and other malicious programs. Another risk is that, if the attacker can add his own public key to the list of trusted CA, he can issue his own certificates, which will be managed exactly like legitimate certificates. They can even match legitimate certificates in every other field except that they would contain a public key of the attacker instead of the trusted user. PKI is not the only solution to produce a secure connection between the client and server; alternatives include Secure Remot Protocol (SRP) and Pre-Share Key (PSK).

2.3.3 Secure Remote Protocol (SRP) Authentication

Secure Remote Protocol (SRP) is a good secure password that is used by most of Internet protocol. SRP produces a password authentication and key-exchange protocol, which is suitable for authenticating users and exchanging keys over unsecured network. In SRP, user passwords are not stored at the server as a plain text, hence if attacker compromise the data in the server, the attacker cannot use them. Also, the legitimate user does not transmit the password in plain text. SRP provides a perfect protection of the past session and passwords against future compromises [17].
Figure 2.6: Overview of Public Key Infrastructure (PKI). Bob applies to Registration Authority (RA) to obtain a digital certificate for his web store. The RA confirms Bob’s identity and contacts with CA to issue the certificate. When the web store visitors present the certificate to Validation Authority (VA), Bob’s identity can be confirmed.

However, there are some disadvantages that prevent the use of SRP. It depends on passwords to provide a client-side authentication, which is one of the human authentication methods. This leads SRP to have the same limitations of the password. For example, SRP does not provide strong non-repudiation, which is one of the most required properties of strong authentication method. Therefore, users can deny the authentication that happened by claiming that someone else used his password.

2.3.4 Pre-Shared Key (PSK) Authentication

A pre-share key is a shared secret key that is shared between two parties using a secure tunnel before using that secret. PSK usually relies on symmetric key cryptographic algorithms. There are three sets of new ciphersuites for the SSL/TLS protocol to support
authentication based on PSKs as specified in [18]. These PSKs are shared in advance through the secure communications between the two parties. The first set uses symmetric key operations only for authentication. The second set of ciphersuites uses a Diffie-Hellman exchange authenticated with PSK. The third set combines the public key authentication of the server with PSK authentication of the client.

Using PSK provides some advantages. The first advantage is that, by using PSK, we can avoid the need of the public key operations because it depends on the ciphersuite. The second advantage is that PSK is more convenient from a key management point of view. For example, when the client and server already have a technique for setting up a shared secret key, that technique could be used to ”bootstrap” a key for authenticating a TLS connection [18]. However, since PSK depends on a shared secret, which is usually a password, it has most of the password limitations.

PKI, SRP, PSK and other methods are used to provide a user authentication in web application. These methods of user authentication helped to make authentication easier for most of the users. However, the web applications are vulnerable to attack, since PKI, SRP and PSK rely on password based authentication, as mentioned above.

2.4 User Authentication in Web Applications

2.4.1 Overview of User Authentication using Web Applications

A web application is defined as a dynamic HTTP-based application whose accomplishes one or more tasks over a network, typically through a web browser [21]. Web applications are popular due the convenience of using a web browser as a client. Unfortu-
nately, most web applications use a weak authentication scheme which is more vulnerable to attack. Web application authentication is critical because providing authentication secure against adversaries is important to both the clients and systems. Unfortunately, some of the web application would provide an easy authentication scheme to provide a better user experience. However, these web applications will be vulnerable by most of the network attackers.

### 2.4.2 Authentication Methods in Web Applications

In the web application, there are three ways of authentications: client-side authentication, server-side authentication or both authenticate each other in the same time. This means a client authenticates itself to a server, the server authenticates itself to the client or in such a way that both parties are assured of the others’ identity which is called mutual authentication. There are many mechanisms available for web application authentication: HTTP Basic authentication, HTTP Digest authentication, HTTPS authentication, Form based authentication, One Time Password (OTP) or Hard Token..etc.

The first two mechanisms are too weak and we talk about them in subsections 2.2.1 and 2.2.2. HTTPS authentication is a strong authentication mechanism, but also it has some limitations; it requires a digital certificate and it commonly is used in e-commerce, e-government and health care web applications. This type of authentication explained in section 2.3. The last three mechanisms will be illustrated in more detail below.

Form Based Authentication and HTTP Basic authentication is the same which are the most common type of web authentication. In this type of web authentication, the server displays a form that ask the user to enter the user name and password for the first time
that the user want to access a secure website. Then, if the user is authenticated, he will be redirected to the protected website; otherwise, he will get a message stating that he/she does not provide a correct user name or password or stayed logged in to the same website for too long, see Figure 2.7. This mechanism has the same lack of security because the user password is sent through the Internet as plain text, exactly like Basic authentication. Some of these problems can be alleviated by adding more protection, such as using SSL/TLS as a secure transport mechanism.

![Diagram](image)

**Figure 2.7:** In Form Based Authentication, the user sends his/her password as plain text through network and the server accepts or rejects him/her depending on the correctness of user’s password.

**OTP** is a type of the multifactors authentication. It is like form authentication, but when the user send his credential data, the server validates the data, then generates a OTP and sends it as a text message, e-mail, or any other method of out-of-band communication. After that, the user uses that message to complete his authentication. Any of these OTPs can be used only once.

**Hard Token** is also a kind of multifactor authentication where the user has a physical token and uses a password at the same time. The user has to enter the number displayed on
the token along with the password that he has in the application. If both values are correct, the user can access to the application; otherwise, he will be rejected.

The SSL/TLS’s security services of integrity, confidentiality and server-side authentication are used by HTTPS’s web applications. However, the SSL/TLS’s client-side authentication is rarely used because of some usability, deployment, and accessibility obstacles which are mentioned above. Examples of web applications that use user authentication include e-government, health care, online banking and all web applications that deal with confidential data.

2.4.3 Limitations of the Current Web Authentication Methods

Web authentication is a practical problem of user acceptability, performance and deployability [22]. These three problems are the main obstacles to providing strong user authentication. As most sites want to increase their visitors, the client authentication should not be that difficult. Users will be frustrated by authentication schemes that ask for more effort on their part, such as installing a certificate or using a specific browser. At the same time this kinds of authentication will not provide a good performance, since the user might make mistakes during authentication.

Security in general costs more in performance. All the service site owners would like to respond to as many orders as they can to make the user happy. However, when the user authentication becomes more secure, the server performance generally degrades. Therefore, having a secure authentication and high performance are difficult perform at the same time.
The last problem of secure web authentication is the deployment. To have a secure web authentication over the Internet, we sometimes need to use a new technologies. These technology are not always available to the user, such as biometrics technologies(i.e., irises scanner, a smart card reader). As a result, we cannot get a secure web authentication everywhere all at once, but some good methods that are worth developing to produce a secure web authentication.
Chapter 3

RELATED WORKS

3.1 Biocryptographic Key Infrastructure (BKI)

The PKI was proposed to protect the data between the client and server from attack. In the PKI, users are not bound strongly with their cryptographic keys. Because the keys are protected by a password that has its weaknesses, such as the ability to be shared or stolen, and also it can be used without the owners’ knowledge. Therefore, by using these keys, we cannot say the users really establish their identities. In response, the BKI integrates biometric authentication capabilities into the traditional X.509 digital certificates to enhance the authenticity of usage of PKI. The BKI is the only solution that provides biometric authentication capabilities into the PKI while still respecting the privacy of biometric data. Unlike previous proposals of integrating biometrics into the PKI’s X.509 certificates, such as [27] [28], these proposals do not provide enough safeguards around biometric data. They store and match unprotected templates or easily invertible representations of individual’s
biometric data. Also, they did not provide a process to re-issue or revoke the biometric template.

### 3.1.1 Biocryptographic Key Infrastructure types

The BKI provides two methods of authenticating a remote user with a BKI certificate. The first type is the biocryptographic system, also known as biotoken which is a securely transformed biometric template. Biocryptographic system depends on the biocryptographic system that deploys the biometrics and cryptography together to provide privacy-enhanced biometric authentication capabilities. As shown in figure 3.1, the secure biotoken is attached to the user’s BKI certificate to provide a remote user authentication by binding a challenge to the biotoken and requesting the user to use his biometrics to release the challenge. The biocryptographic system sends the individual’s biometric as a cryptographically secure biometric template, which preserve the privacy of individual’s biometrics while providing the same strong biometric authentication. Figure 3.1 illustrates the process of how the cryptographic key binds with a biotoken and how it will be released after successful biometric matching. Also, a passcode can be added to the biotoken to provide two-factor security, which increases privacy protection. Hence, to release the cryptographic key, we have to use the biometric and password.

The second type uses the BKI’s biometric public key, which is a part of the BKI’s Asymmetric BioCryptographic Subsystem(ABCS) [25]. This type proposes asymmetric encryption/decryption operations with necessity of using biometric authentication to use the biometric private key. ABCS provides four different approaches that help to access the bioprivate key in a secure way. First approach, the bioprivate key could be kept inside a
Figure 3.1: The BKI certificate format. The BKI certificate is leveraged by BSSL in order to perform biometric client-side authentication. [25]

(a) Overview of the key binding process in the BKI

(b) Overview of the key release process in the BKI

Figure 3.2: Cryptographically secure biotoken overview. More detailed information can be found in [29].

trusted biometric sensor. The second It could be encrypted with a key that is kept inside a trusted biometric sensor. Third approach is either of above approaches could use any Trust execution Environments (TEE). The last approach is providing a software that securing the bioprivate key.
(a) The biocryptographic system is used to release the challenge from the biotoken.  

(b) The BKI’s ABCS is used with the biometric private key to releasing the challenge.

Figure 3.3: Alice and Bob participate in this protocol to authenticate each other. Each of them can only obtain the other challenge after biometrically authenticating himself/herself. The authentication is performed to either

Our implementation used the last approach which is a software model (i.e., using a key binding biocryptographic system). The BKI’s ABCS could use the biometric sensors with secure on-chip matching to produce a remote biometric authentication. For instance, the biggest smart phones tablets and laptops manufacturers deploy a fingerprint sensors in their products (i.e., Samsung Galaxy S5 and Apple iPhone 6). However, these sensors cannot be used for remote biometric authentication because they do not release the biometric library for public to use. Therefore, the BKI’s ABCS can be used on these sensors if their owners allow us to use them by releasing the biometric library. As a result, the web browser in smart phones or laptops can authenticate the user using the biometric sensors with secure on-chip biometric matching. Figure 3.3 shows the remote biometric authentication protocol using either the biotoken or BKI’s ABCS. Also, it shows that the time stamps and cryptographic nonces are used to ensure that their messages are fresh and as a countermeasures against an attacker’s reply.
3.2 The Biocryptographic Secure Socket Layer (BSSL)

Albhdal and Boult proposed a BSSL [24], which aims to enhance security and usability of remote authentication with the integration of biocryptographic authentication capability into SSL/TLS handshake protocol in a usable, secure, and privacy-preserving way. The BSSL proposal aims to enhance the security by introducing the biometric client-side capabilities instead of using a password, which does not provide strong authentication or non-repudiation and is also vulnerable to a wide range of attacks. Moreover, the BSSL proposal enhances the usability aspect by not requiring the user to memories long and complicated passwords for multiple systems. Instead, users only have to submit their biometric (i.e., face, iris or fingerprint), which are always on hand for users.

3.2.1 The Biocryptographic Secure Socket Layer Types

As we mentioned above, BKI provides two types of a remote user authentication. Therefore, the BSSL uses both methods to introduces two different types of BSSL full handshake protocols with biometric client-side authentication. In our proposal, we concentrate on the second type of BSSL handshake protocol, which relies on BKI’s ABCS. Figure 3.4 shows the sequence of the BSSL full handshake protocol using the BKI’s ABCS. The main advantages of BSSL handshake protocol are that 1) it uses the same messages of the SSL/TLS handshake protocol, and that 2) it can be deployed with minimal modifications to the SSL/TLS handshake protocol.

HTTPS relies on SSL/TLS to protect the data between the client and server. Therefore, we will depend on BSSL to produce a new protocol which is Hypertext Transfer
Protocol Biocryptographic (HTTPB). The following chapter will discuss HTTPB protocol in more details.

3.2.2 BSSL ABCS Handshake agreement

As shown in figure 3.4, the client begins the BSSL handshake agreement using the BKI’s ABCS by sending a hello message to the server. The client hello message includes a TLS extension field that shows the client is willing to use BSSL client-side authentication with the BKI’s ABCS. If the server support the BKI’s ABCS handshake protocol, the server reacts with a server hello message that includes a TLS extension that confirms use of the BSSL handshake agreement with the BKI’s ABCS. At that point, the server sends its certificate, the server key exchange, certificate request, and the server hello message done. When the client receives the server hello message done, the client reacts by sending his certificate message, which includes the client BKI certificate. The client certificate is sent alongside a client key exchange message and a certificate verify message that includes the client signature using his biometric private key on the digest of all the handshake messages that were previously exchanged. This signature is used for client confirmation since only the client can access his/her biometric private key, which is secured using the BKI’s ABCS. At that point, the client sends the server the change cipher spec and finished messages to inform the server that the client has finished the handshake agreement and began the protected connection. After the server gets the client BKI certificate and certificate verify messages, the server verifies the certificate verify message by confirming that the signature appended to it utilizing the biometric public key, which was appended to the client biomet-
If the signature is true, the server considers the client valid and sends the client the change cipher spec and finished messages and begins the safe connection.

Figure 3.4: BSSL full handshake protocol with biometric client-side authentication relying on the BKI’s ABCS. Messages labeled with asterisks (*) are optional messages [24].
Chapter 4

THE HYPERTEXT TRANSFER PROTOCOL

BIOCRYPTOGRAPHIC (HTTPB) DESIGN AND IMPLEMENTATION

4.1 Overview of the HTTPB’s Design and Implementation

The HTTPS protocol is based on the Secure Sockets Layer (SSL) or the Transport Layer Security (TLS) to secure the Hypertext Transfer Protocol (HTTP) web communications. Both the SSL and TLS protocols use the Public Key Infrastructure (PKI). In turn, the PKI facilitates the exchange of keys between the client and server. These keys are not bound strongly to their owner. In response to this limitation of the SSL/TLS client-side authentication, we implement the Biocryptographic Secure Socket Layer (BSSL) on Mozilla Network Secure Services (NSS) library. The previous proposal of the BSSL [24] is implemented in OpenSSL library, which is one of the common implementation of SSL/TLS. BSSL protocol is illustrated in more detailed information in the Related Work chapter. As shown in figure 4.1, Hypertext Transfer Protocol Biocryptographic (HTTPB) uses
BSSL and BSSL while relying on the Biocryptographic Key Infrastructure Asymmetric Biocryptographic subsystem (BKI’s ABCS) [19] [25]. The following section will discuss the HTTPB in more details.

4.2 Hypertext Transfer Protocol Biocryptographic Design

The HTTPB provides the same security services of HTTPS with the addition of the strong biometric-based client-side authentication capabilities provided by BSSL. Simply, the HTTP will be used over BSSL precisely as using HTTP over Transmission Control Protocol (TCP). Hence, HTTPB relies on top of BSSL, as shown in figure 4.2, in the same way that HTTPS relies on SSL. The user acting as the BSSL client must also act as the HTTP client. It should initiate a connection to the server and send the BSSL client hello message which is the first data that the BSSL server expects to receive to begin the BSSL
handshake. When the BSSL handshake has finished, the client may then initiate the first HTTP request. All HTTP data must be sent as BSSL "application data". Consequently, when HTTP/BSSL is being run over a TCP/IP connection, the default port is 443. In addition, both parties should identify each other to prevent the man-in-the-middle attacks.

BSSL/HTTPB aims to overcome the aforementioned security and usability problems of SSL/HTTPS client-side authentication. BSSL/HTTPB integrates biometrics authentication capabilities into SSL/HTTPS in secure, and privacy-preserving manner. As SSL/HTTPS relies only on Public Key Infrastructure (PKI), our design of HTTPB depends on the (BKI)’s ABCS and its biometric certificates.

Moreover, the user can store his BKI’s certificate at the server side and send a URL that pointers where the user certificate is hosted. Thus, when the user start the BSSL ABCS handshake the client and the server uses the TLS extensions, in the client and server hello messages, to tell each other that they are going to use the URL certificate. According to the [24]’s experiments, they compared sending the client certificate during the handshake with handshake protocol using certificate URLs, The experiment shows that sending a certificate URL introduced, on average, 25% overhead compared to sending the client certificate during the handshake time. Even though, we consider using client certificate URLs BSSL handshake protocols as a practical and usable solution to the usability problems of HTTPS/SSL client-side authentication. Therefore, storing client BKI certificate at the server side will provide more usability.
4.3 HTTPB Implementation

To implement the HTTPB, we have to implement the BSSL handshake protocols first by extending the NSS library implementation of SSL/TLS because the HTTPB relies heavily on BSSL protocol and because we are going to use the Firefox web browser, which relies on NSS library to provide secure communication. We extended the TLS 1.2 handshake protocol of NSS library to support the BSSL handshake protocols using BKI’s ABCS.

In our implementation, the user should have a biometric digital certificate and biometric private key in order to authenticate himself/herself using his/her biometrics. The user has to convert his biometric certificate to (pfx) type, which is only the Firefox accepts as a client certificate type. Then the user has to upload the certificate to Firefox in order to use it during the handshake time. Therefore, when Firefox starts a connection using the HTTPB protocol, it uses the user digital certificate that was uploaded earlier. Also, the HTTPB requests the user fingerprint during the handshake time. If the print does not match with the one embedded with the bioprivate key, the handshake will not be completed; otherwise, the handshake will complete normally. In this manner, the Firefox web browser
uses the BSSL’s ABCS handshake protocol to complete the secure connection between the client and the server, assuming the server supports the BSSL’s ABCS handshake protocol as well.

On the server side, we enhanced the Apache web server to support the HTTPB/B-SSL ABCS handshake protocol. We reconfigured the Apache web server and made it use the OpenSSL that provides a secure implementation of BSSL ABCS and a wide range of cryptographic primitives. Also, creating a self-signed biometric certificate was one of our jobs to provide a biometric digital certificate to the Apache web server in order to be used during the handshake time with the client.

### 4.4 Experimental Results

In order to examine the execution of our implementation, we performed an experiment. The point of this experiment is to guarantee that the HTTPB protocol is usable and practical regarding the time required in finishing the handshake between the client and the server. For this purpose, we rely on the Apache web server that has the OpenSSL library which supports the HTTPB/BSSL ABCS and HTTPS/TLS 1.2 standard at the same time. We used a PC equipped with an Intel Core i5 processor (2.6 GHz) and 8GB of memory. We ran the client and the server in the same machine to take out the system delay and investigate the computational overhead of HTTPB protocol by contrasting it with the HTTPS. We ran our implementation HTTPB/BSSL ABCS and the HTTPS/TLS 1.2 at least 30 times. Then we got the average handshake time that is consumed for both of them to make our comparison.
Figure 4.3: The average total time that required to perform the HTTPB and HTTPS protocols using the same machine.

Figure 4.3 shows the average time required to perform the HTTPB protocol compared to HTTPS protocol when the client and server are hosted in the same machine. As a result, the HTTPB introduce significantly higher computational overhead compared to HTTPS. The reason is that the HTTPB has expensive operations of encoding of the biotoken. This encoding operation happens when the client needs access to the biometric private key. Despite the fact that HTTPB protocol has introduced higher computational overhead, we did not experience any handshake that exceeded 2 seconds.

In addition, Albahdal et al. [30] measure the efficiency of password and fingerprint biocryptographic authentication for both the enrollment and verification processes. For the password enrollment, they measure the total time needed to successfully type a strong password and its confirmation. For the password verification, they measure the required time to successfully type a password that matches the enrollment’s password. For fingerprint biocryptographic enrollment, they measure the required total time to successfully scan the
user’s fingerprint three times using a fingerprint scanner. After that, they measure the total time required for fingerprint biocryptographic verification to successfully scan the user’s fingerprint where the new fingerprint reading matches the enrollment’s fingerprint reading (i.e., biocryptographic matching). The experimental results as shown in Figure 4.4, state that the fingerprint biocryptographic authentication is faster than the password authentication. The fingerprint biocryptographic verification process is more than 3 times faster than the password verification process. Therefore, we conclude that HTTPB protocol is usable and practical when the usability and security features that HTTPB provide are considered.

Figure 4.4: Average password typing time versus fingerprint scanning time for enrollment and verification processes. [30]
Chapter 5

FUTURE WORK AND CONCLUSION

5.1 Future Work

While our implementation is one of the first steps to enhancing security and usability in the HTTPS/SSL, there are some additional points to be completed to make our end goal more usable and practical. For example, we implement only the BSSL ABCS protocol in the NSS library while there is another type of BSSL protocol, the BSSL Biotoken. Implementing the second type of BSSL protocol gives the user more choices for remote user authentication. Also, we used the Firefox web browser to implement the HTTPB protocol. Thus, this implementation can be done in another popular web browser such as Google Chrome or Internet Explorer. Moreover, we hope to allow the usage of secure on-chip matching sensors for remote application (i.e., Iphone 6 and Samsung Galaxy S5). On-chip matching sensors provide privacy without downloading the matching accuracy (i.e., the privacy vs. matching accuracy trad-off).
5.1.1 Operating the HTTPB in Real Life

Since we have seen the basic framework and protocols, we can start to consider the utility of HTTPB for the web browsers. To apply HTTPB in real life, some steps should be taken from both the client and server sides.

On the client side, the user should have a biometric private key embedded in a fingerprint scanner. These scanners should be issued by a third trusted party, Certificate Authority (CA). Hence, the client should request a biometric digital certificate from the CA. This type of scanners helps the legitimate user to authenticate himself/herself by using his/her fingerprint during the client-side authentication which will provide privacy without downloading the matching accuracy.

On the server side, the web server system should be improved to accept and deal with the BSSL ABCS handshake protocol. Thus, when the handshake protocol starts between the client and server, both sides will know how to deal with the other party using BSSL ABCS protocol. We improved the Apache web server to deal with BSSL ABCS protocol or SSL/TLS protocol in the same time, and the same could be done for other popular web servers have to include the BSSL ABCS handshake protocol, such as Microsoft Internet Information Services (IIS) and Sun Java System.

5.1.2 Learning and Experience

During my research, I met some challenges that improved my abilities and enabled me to reach my goal. These challenges include compiling and debugging the FireFox source code; linking the Network Secure Services (NSS) library with other libraries; and
finding out how the SSL handshake protocol is implemented in the NSS library and what the differences are between OpenSSL and NSS libraries.

While editing the Firefox source code, I found that developing and editing others’ code is much harder than editing my own, or even generating a new code altogether. The difficulty of editing depends on several factors: how the previous code was organized, how large is it, which language is used and who implemented it. Firefox is one of the biggest projects I have ever seen. Therefore, it was a challenge to find a way to get into it and locate where to add the changes. Moreover, I had to decide what the best tools are that I have to use, and what is the best editing environment that I am familiar with.

I used Microsoft Visual Studio under Microsoft Windows 7 because I have some experience using it. Hence, the first thing I was thinking about how to debug the code and follow it line by line to find out where I have to add my code. This step took a lot of time until I found an appropriate method that helped me debug the Firefox web browser in a proper way.

Another challenge that I met was making a link between NSS library and another libraries such as OpenSSL [26], Biotope [29] and BKI [19]. All these libraries were linked with NSS library so they could be used in our implementation.

The previous proposal of BSSL was implemented in OpenSSL library. Hence, I studied that proposal for a long time to understand the idea of the BSSL implementation on OpenSSL, so I could attempt to implement the same thing on the Mozilla NSS library. As a result, I found so many differences between OpenSSL and NSS that it took longer to figure out where I needed to apply my changes in the NSS library; while both libraries (OpenSSL and NSS) perform the same job, they have their own SSL/TLS’s implementation.
Providing self-signed digital certificates was important, to the point where it was practically mandatory, which was another challenge. I used OpenSSL library to provide a self-signed biometric digital certificates. Then I converted them from (pem) type to (pfx) type, which is the only form accepted by Firefox as a digital certificate.

All these challenges, and many others, prevented me from finishing earlier in the Spring semester 2015. However, I believe that these challenges helped to make my research strong enough to be a master thesis.

5.2 Conclusion

The communication link between the client and the server is usually protected against eavesdropping using SSL/TLS. Despite these protective implementations, some security threats remain. One reason for these remaining vulnerabilities is that SSL/TLS does not obligate client authentication. As a result, it is not easy to verify if the person who is making a payment is the legitimate cardholder or not. Consequently, a way to reduce this risk is to perform user authentication. On the one hand, failure to authenticate a transacting party properly may lead to situations such as unauthorized ordering of goods, illegal transfer of funds, and repudiation of completed transactions. On the other hand, complex authentication systems may sacrifice the convenience of online services and render them unusable. With all of these considerations in mind, we introduce secure and privacy-enhanced biometric client-side authentication to the HTTPS. We implemented the Biocryptographic Secure Socket Layer Asymmetric BioCryptographic Subsystem (BSSL ABCS) on the Mozilla Network Secure Services (NSS) library. BSSL ABCS integrates biometrics authentica-
tion capabilities into SSL/TLS in a usable, secure, and privacy-preserving manner. By using BSSL ABCS, we extend the implementation of the Mozilla Firefox web browser to support the Hypertext transfer protocol Biocryptographic (HTTPB). Therefore, we used a fingerprint scanner to get a fresh user biometric in our implementation to authenticate him/her.

We also developed the Apache web server and changed its configuration to support the BSSL ABCS protocol. The Apache web server is now reconfigured to accept and deal with the clients who use the HTTPB protocol.

Also, our experiment shows that the HTTPB protocol is usable and practical in terms of the time that is used in completing the handshake between the server and client when compared with HTTPS protocol.
References


