

# **A Framework Extending TLS/SSL Level Security To The Application Layer**



By

**Zahoor Ahmed Alizai**

**Fall 2017-MS(IS) - 00000205348**

Supervisor

**Dr. Hasan Tahir**

**Department of Computing**

A thesis submitted in partial fulfillment of the requirements for the degree  
of Masters of Science in Information Security (MS IS)

In

School of Electrical Engineering and Computer Science,  
National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

(January 2020)

# Approval

It is certified that the contents and form of the thesis entitled “**A Framework Extending TLS/SSL Level Security To The Application Layer**” submitted by **Zahoor Ahmed Alizai** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Hasan Tahir**

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Committee Member 1: **Dr. Seemab Latif**

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Committee Member 2: **Dr. Yousra Javed**

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Committee Member 3: **Ms. Hirra Anwar**

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

# Thesis Acceptance Certificate

Certified that final copy of MS/MPhil thesis written by **Mr. Zahoor Ahmed Alizai** (Registration No **Fall 2017-MS(IS) - 00000205348**), of SEECS has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Advisor: **Dr. Hasan Tahir**

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Head of Department (HoD)

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Dean/Principal

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

# Dedication

I dedicate this thesis to my **Parents** for their endless prayers, love and encouragement.

# Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Zahoor Ahmed Alizai**

Signature: \_\_\_\_\_

# Acknowledgment

First of all, I would like to thank Allah, the Almighty for giving me the ability and strength to carry out this research.

My deepest gratitude to my supervisor Dr. Hasan Tahir for his continuous support and guidance during my thesis. I could not have imagined having a better supervisor and mentor for my master's degree. I am also thankful to my teachers for providing me with an academic base, which enabled me to complete this thesis.

I am thankful to all my fellows and friends especially Malik Hamza Murtaza and Mohammad Taha Ajani for their support and motivation.

Last but not the least, I would like to thank my parents for their endless prayers and support throughout.

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Overview . . . . .	1
1.2	Problem Statement . . . . .	4
1.3	Solution Definition/Description . . . . .	4
1.4	Thesis Motivation . . . . .	5
1.5	Thesis Contribution . . . . .	5
1.6	Thesis Organization . . . . .	6
1.7	Summary . . . . .	7
<b>2</b>	<b>Literature Review</b>	<b>8</b>
2.1	OSI Model . . . . .	8
2.1.1	Physical Layer . . . . .	9
2.1.2	Data Link Layer . . . . .	9
2.1.3	Network Layer . . . . .	10
2.1.4	Transport Layer . . . . .	10
2.1.5	Session Layer . . . . .	11
2.1.6	Presentation Layer . . . . .	11
2.1.7	Application Layer . . . . .	12

*TABLE OF CONTENTS* vii

2.2	Authentication Techniques . . . . .	12
2.2.1	Password-based Authentication . . . . .	13
2.2.2	Key-based Authentication . . . . .	15
2.2.3	Multi-factor Authentication . . . . .	16
2.2.4	Biometric Authentication . . . . .	17
2.2.5	Hardware Authentication . . . . .	18
2.3	Session Management . . . . .	19
2.4	Cookies and Tokens . . . . .	21
2.5	Transport layer Security Mechanisms . . . . .	23
2.6	Summary . . . . .	24
<b>3</b>	<b>Research Methodology</b>	<b>25</b>
3.1	Introduction . . . . .	25
3.1.1	Research Types . . . . .	26
3.1.2	Research Methods and Research Methodology Overview	28
3.2	Thesis Research Methodology . . . . .	29
3.3	Summary . . . . .	30
<b>4</b>	<b>Proposed Solution</b>	<b>31</b>
4.1	Scheme Primitives . . . . .	31
4.1.1	Timestamps . . . . .	31
4.1.2	Random Numbers . . . . .	32
4.1.3	Symmetric Cryptography . . . . .	32
4.1.4	Asymmetric Cryptography . . . . .	33
4.1.5	Hash Functions . . . . .	33
4.1.6	Message Authentication Code (MAC) . . . . .	34



4.2	Scheme Features . . . . .	34
4.3	Proposed Solution . . . . .	35
4.3.1	Registration Phase . . . . .	36
4.3.2	Authentication Phase . . . . .	38
4.3.3	Session Key Establishment . . . . .	38
4.3.4	Scheme Handshake . . . . .	39
4.4	Security Provisions . . . . .	40
4.5	Summary . . . . .	41
<b>5</b>	<b>Implementation and Results</b>	<b>42</b>
5.1	Scheme Analysis . . . . .	42
5.2	Attributes . . . . .	43
5.2.1	Secrecy . . . . .	43
5.2.2	Aliveness . . . . .	43
5.2.3	Synchronization . . . . .	44
5.2.4	Man-In-The-Middle . . . . .	44
5.3	Scyther Script . . . . .	44
5.4	Scyther Results . . . . .	46
5.5	Testing and Implementation . . . . .	47
5.5.1	Environment Setup . . . . .	48
5.5.2	Functionality . . . . .	54
5.5.3	Results . . . . .	55
5.6	Practical Adoption/Case Study . . . . .	55
5.7	Summary . . . . .	58

<i>TABLE OF CONTENTS</i>	ix
<b>6 Conclusion &amp; Future Work</b>	<b>59</b>
6.1 Conclusion . . . . .	59
6.2 Future Work . . . . .	60
6.3 Summary . . . . .	61
<b>Bibliography</b>	<b>62</b>
<b>Appendix A Scyther</b>	<b>73</b>
<b>Appendix B Java</b>	<b>77</b>
<b>Appendix C PHP (Server-side)</b>	<b>102</b>
<b>Appendix D JavaScript (Client-side)</b>	<b>116</b>

# List of Figures

1.1	Standard Software Development Lifecycle . . . . .	2
2.1	Physical Layer . . . . .	9
2.2	Data Link Layer . . . . .	9
2.3	Network Layer . . . . .	10
2.4	Transport Layer . . . . .	10
2.5	Session Layer . . . . .	11
2.6	Presentation Layer . . . . .	11
2.7	Application Layer . . . . .	12
3.1	Research Types . . . . .	26
4.1	Proposed Scheme . . . . .	36
4.2	Sample Certificate . . . . .	37
4.3	Scheme Handshake . . . . .	40
5.1	Scyther Script - 1 . . . . .	45
5.2	Scyther Script - 2 . . . . .	45
5.3	Scyther Script - 3 . . . . .	45
5.4	Scyther Script - 4 . . . . .	46

*LIST OF FIGURES*

xi

5.5	Scyther Script - 5 . . . . .	46
5.6	Scyther Result . . . . .	47
5.7	Client Server Application (JAVA) . . . . .	49
5.8	Traffic Capture - Wireshark . . . . .	49
5.9	PHP - 1 . . . . .	50
5.10	PHP - 2 . . . . .	51
5.11	PHP - 3 . . . . .	51
5.12	PHP - 4 . . . . .	52
5.13	Traffic Capture - Burp Suite - 1 . . . . .	52
5.14	Traffic Capture - Burp Suite - 2 . . . . .	53
5.15	Traffic Capture - Burp Suite - 3 . . . . .	53
5.16	Traffic Capture - Burp Suite - 4 . . . . .	54

# List of Tables

4.1	Table of Notations . . . . .	36
5.1	Tools & System Specifications . . . . .	48

# List of Publications

1. Z. A. Alizai, N. F. Tareen, and I. Jadoon, “Improved IoT Device Authentication Scheme Using Device Capability and Digital Signatures,” in *2018 International Conference on Applied and Engineering Mathematics (ICAEM), 2018*, pp. 1–5.
2. Z. A. Alizai, H. Tahir, M. H. Murtaza, S. Tahir, and K. Mcdonald-Maier, “Key-Based Cookie-Less Session Management Framework for Application Layer Security,” *IEEE Access*, vol. 7, pp. 128544–128554, Sep. 2019.
3. M. H. Murtaza, Z. A. Alizai, and Z. Iqbal, “Blockchain Based Anonymous Voting System Using zkSNARKs,” in *2019 International Conference on Applied and Engineering Mathematics, ICAEM 2019 - Proceedings, 2019*, pp. 209–214.

# Abstract

By design Transport Layer Security (TLS) and Secure Socket Layer (SSL) are not designed to work at the application layer. This means that there is considerable security protocol isolation between the upper and lower layers of the network stack. The core objective of this research is to extend security features of TLS and SSL to the application layer. The proposed solution intends to bind multiple security features such as authentication, mutual authentication, continuous authentication, and session management in a single secure scheme thereby ensuring that application developers do not have to deal with security implementations as the same is provisioned through the proposed scheme. The proposed scheme will embed security mechanisms like access control and group authentication on top of the extended security provisions. Thus, improving the overall security of the system drastically. Session management and authentication is achieved using asymmetric keys without the use of session cookies and session tokens thus mitigating attacks such as cookie theft, token forgery and nullifying a vast group of attack vectors. The proposed scheme has been implemented and tested for security conformance thereby proving its effectiveness and practicality.

# Chapter 1

## Introduction

Chapter 1 elaborates the overview of basic concepts, significance and history of research work. This chapter describes the road map for our thesis and briefly highlights the further organization and structure of the thesis. This chapter explains the motivation for carrying out the research work. This chapter also gives idea about the vital contributions, scope of the work and key objectives of the thesis.

### 1.1 Overview

Technological advancements have allowed us with the ease of information systems being available on mobile devices. Education, banks, health, etc. [1, 2], all of whom have experienced significant improvement in remote access. As availability of services and complexity increase, the risk of malicious factors compromising the information systems also increases. The information system at risk can have serious consequences, ranging from nuclear meltdowns



to identity theft [3]. Security systems that are based on Transport Layer Security (TLS) [4] and SSL [5] are secure. But these traditional protocols provide security to moving data; however, these protocols do not secure data, which is used by information systems, *i.e.* data at rest. Therefore, TLS and SSL fail to give security assurances to the systems which are subject to attacks that are not prevented by TLS and SSL. Provable security is provided by TLS when transferring data [4]. However, some applications are depended on sessions that are at the application layer which are different from the sessions at the transport layer. Serious design flaws can occur in applications due to the usual practice of developing of application session mechanisms by application developers with little or no security knowledge [6]. As shown in figure 1.1, security is often considered a requirement and is not part of the conventional Software Design Lifecycle (SDLC) [7].

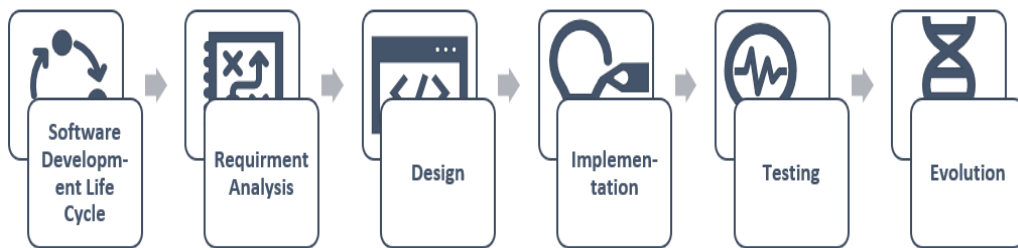


Figure 1.1: Standard Software Development Lifecycle

Unification of session mechanisms in the application layer is rare and readily available frameworks are lacking [8]. Application development remains very sensitive to issues such as incorrect coding, logical errors [9] etc. because of these issues. It is difficult to make a security system which not only provides security at the transport level but also it provides security to the data that is available at the application layer. Since there is a diversity

in information systems, this makes it hard to set up a unified security architecture. The practice of implementing customized security measures at the application layer by the application developer often results in catastrophic failures [8].

Correct functioning of the information system depends on the authentication phase. Flawed authentication may result in provisioning of resources to an adversary or lack of provisioning to a legitimate user. Problems at the authentication phase often result in systems being compromised which has been well documented in the literature [10]. When the user is first authenticated, it is necessary for the system to continue to authenticate consecutive requests that come to the system without explicitly cancelling the overall process of authentication. This process is called session management [11]. The most important phase where a system is usually compromised is at session management phase, for example, cookies and tokens are the main sources of continuous authentication for a web system, but they can be attacked and are a reason for increased attack surface. The analysis of existing protocols and mechanisms show that standard for application-level session management is lacking, while transport-level session management is widely standardized [8]. The disparity between these layers means that the application layer cannot take advantage of session management at transport layer. If sessions are extended to the application layer, the efforts of securing the system will be reduced considerably, and it will provide better security to the systems.

Authentication information is collected at the registration step in many existing systems. This research focuses on how the key setup or key exchange can benefit from initial registration phase in the registration process.

Many systems benefit from multi-factor authentication, same can be used to simplify key revocation and key setup. As a result of this, we will be able to achieve mutual authentication, which is better than server authentication in conventional setups. As establishing trust between parties who are communicating over unsafe media is a difficult problem to tackle, the proposed scheme can be used in environments where conventional PKI setup is not suitable. Likewise, there are some situations where it is not ideal for transferring trust to the third party; the proposed scheme can be used to build trust [12]. For example, to obtain fake TLS certificates [13], Border Gateway Protocol (BGP) is used, and then these fake TLS certificates are sold on the dark web [14]. A multi-party communication scenario can also benefit from this scheme where participants can be identified to use a resource or in a situation where the identity of the individual is masked by the group.

## 1.2 Problem Statement

The following problem statement has been formulated based on the problems identified during literature review.

“Design and implementation of a comprehensive security framework that will extend the guarantees provided by TLS/SSL implementations at transport layer and apply them to the application layer.”

## 1.3 Solution Definition/Description

The proposed solution provides:

- Session management, authentication and continuous authentication.
- Guarantees built on top of transport layer security mechanisms.
- Elimination of passwords, cookies and tokens.
- Provide basis for security mechanisms, such as access control.

Through the implementation of the proposed framework, the attack surface is drastically reduced.

## 1.4 Thesis Motivation

Conventional security systems secured using protocols like TLS or SSL are considered highly secure. It's a valid assumption; however, it is associated only with data which is in transport mode, and they do not provide security to data in use by the information system. Lack of standardization for application layer security leads to severe security breakdowns.

## 1.5 Thesis Contribution

The main contributions are listed as follows:

- The issue of TLS not extending its cryptographic guarantees to the application layer has been brought to light and the associated concerns.
- Through research it has been shown that cookie theft and token forgery are the cause of security drawbacks of application layer sessions as they are based on cookies and tokens.

- Discussed the implications of a non-existent unified security architecture which should provide cryptographic guarantees on the application layer.
- Studied the design of a cookie and token-less methodologies for authentication, mutual authentication, and continuous authentication based on cryptographically secure primitives. Hence mitigating attacks such as cookie theft and token forgery.
- The proposed design uses cryptographic keys as an alternative to the traditional password-based authentication mechanism.
- Designed a session management scheme that is tightly bound to the authentication mechanism of an application.
- The proposed scheme is tested using a server/client setup implementing the proposed scheme to achieve session management continual authentication and mutual authentication.
- The scheme has been tested by deploying on an http server. An https like functionality is achieved in the absence of PKI. The system did not rely on cookies or tokens for the establishment and continuation of user sessions.

## 1.6 Thesis Organization

The organization of the thesis is presented as follows. Chapter 2 presents the literature review of the important concepts related to this thesis. In chapter

3, the research methodology followed during the research has been discussed. In chapter 4, the proposed scheme and its characteristics are presented. In Chapter 5, a discussion on the testing, implementation, and results of the proposed scheme is presented. Chapter 6 presents the conclusion and proposes future work of the proposed scheme.

## 1.7 Summary

In this chapter, basic concepts and terminologies are introduced regarding network security, transport layer security and application layer security. It gives an overview of aim and scope of the thesis and presents the main objectives of the research work with overall thesis organization. In the next chapter we will look at the literature review that has been conducted for this thesis.

# Chapter 2

## Literature Review

Chapter 2 discusses the related work and terminologies. The related work is basically the research carried out by different researchers over the years which is related to the work done in this thesis and contributed towards making a new solution.

### 2.1 OSI Model

The Open System Interconnection (OSI) is a networking model that helps in the implementation of network protocols in 7 steps/layers [15]. The model explains how control is passed from layer to layer starting from top layer, *i.e.* application layer to the bottom *i.e.* physical layer. Seven layers of the OSI model are discussed briefly one by one below.

### 2.1.1 Physical Layer

The physical layer is the first and bottom layer of the OSI model. This layer is responsible for representing the physical and electrical interpretation of the system. This layer is responsible for conveying the bit-streams, radio or light signal over the network at mechanical and electrical level. Figure 2.1 shows the physical layer.



Figure 2.1: Physical Layer

### 2.1.2 Data Link Layer

The data link layer is the second layer in the OSI model. Data link layer is much like network layer except that it is used in transferring of data between two devices over the same network. This layer is responsible for error control and flow control in the communication. The process of breaking packets into small frames is done on this layer. Figure 2.2 shows the data link layer.

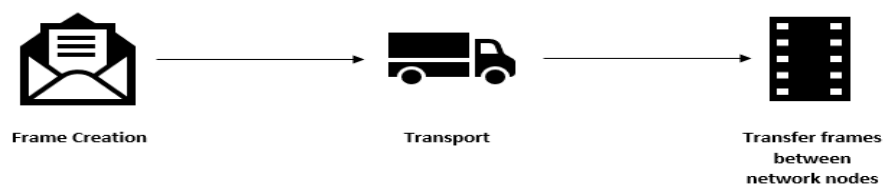


Figure 2.2: Data Link Layer



### 2.1.3 Network Layer

The network layer is the third layer in the OSI model. It is responsible for transferring data between the two networks. The process of breaking segments that came from transport layer into small units called packets is performed on this layer. The responsibility of finding the best physical path for data to reach the destination also lies with the network layer. Figure 2.3 shows the network layer.

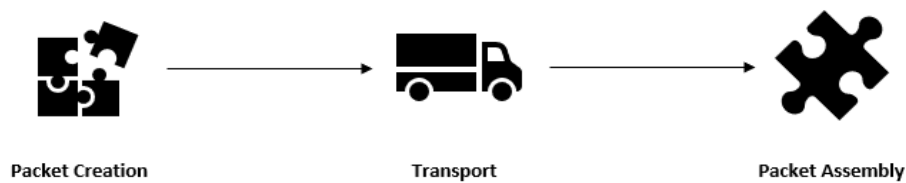


Figure 2.3: Network Layer

### 2.1.4 Transport Layer

The transport layer is the fourth layer of the OSI model. The responsibility of this layer is to facilitate end to end communication between two devices. The process of breaking data which is received from session layer into segments and then sending it to third layer is done by this layer. This layer is also responsible for error and flow control. Figure 2.4 shows the transport layer.

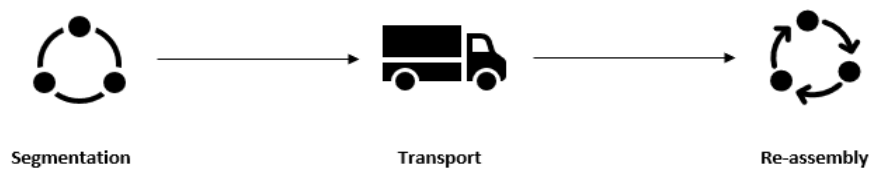


Figure 2.4: Transport Layer

### 2.1.5 Session Layer

The session layer is the fifth layer of the OSI model. This layer has the responsibility of starting and end of communication between two devices. The session is the time between the start and end of the communication. It is the responsibility of session layer to keep the session active until the data is transferred successfully. Figure 2.5 shows the session layer.



Figure 2.5: Session Layer

### 2.1.6 Presentation Layer

The presentation layer is the sixth layer of the OSI model. The main responsibility of this layer is to prepare the data for the application layer. This layer is responsible for compression, translation and encryption of data which is to be presented to application layer. Figure 2.6 shows the presentation layer.

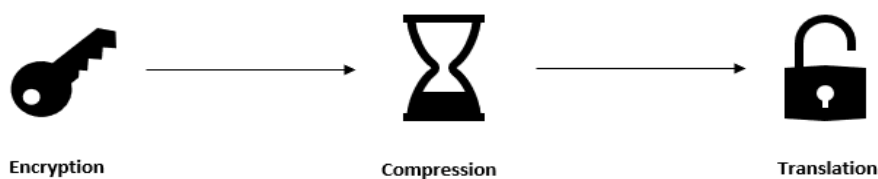


Figure 2.6: Presentation Layer

### 2.1.7 Application Layer

The application layer is the seventh and top layer of the OSI model. It is responsible for providing services to the application program to communicate with another application program effectively over the network. This layer has direct link with the user. All user interactions take place on the application layer. Applications like email clients and web browsers depend on this layer to initiate the communication. Figure 2.7 shows the application layer.

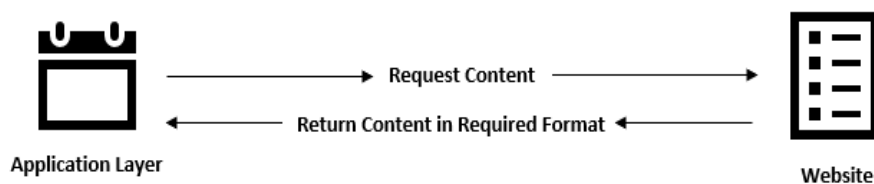


Figure 2.7: Application Layer

## 2.2 Authentication Techniques

When accessing a digital device or service, the most important and first activity is to identify the legitimate operator so that only an authorized user can access the service or digital device. Authentication [16] acts as the main countermeasure that guarantees that a legitimate user has accessed the device or service, and an unauthorized user is prohibited from accessing the device or service. Over the years, many methodologies and techniques have been developed to serve as authentication mechanisms. Web applications [17] which are of distributed nature and inherently stateless, require authentication data to be submitted for each service request. This continuous authentication [16] obstructs the proper functioning of service; hence there

is a need for a methodology that returns the authentication information, *i.e.* the application sessions.

Authentication in any information system is one of the main challenges. Without an infallible authentication procedure, the system remains extremely vulnerable and is prone to many malicious interventions. In simple words, authentication is the verification of an entity whereas authorization is granting access to a verified entity. Some of the main authentication techniques are described below.

### 2.2.1 Password-based Authentication

Password-based authentication is the dominant and reliable authentication mechanism in use. Password-based authentication is easy to deploy; however, it is considered a rapidly aging mechanism for authentication.

In [18], authors have performed analysis of several password authentication schemes and claim that in password authentication schemes tradition of using mathematical operations which are computationally exhaustive can result in security vulnerabilities in the scheme. They have also specified a procedure to crypt-analyze and perform a denial of service attack [19] on those schemes. The first of the four schemes that they have analyzed is by Yang *et al.* [20] which uses passwords in their scheme for authentication and does not use timestamps, and it is used to preserve identity privacy. The second scheme that they analyzed is by Islam [21], it is a smart card-based user password authentication scheme in which timestamps are used. The third scheme is by Jiang *et al.* [22] in which smart cards are not used for authentication.

Fourth studied scheme is by Wang *et al.* [23] which is claimed to be resistant to DoS attacks and is a smart card-based authentication scheme. Authors highlight that strong point of all the schemes, e.g., Yang *et al.* scheme uses encryption and nonce and Islam's scheme uses timestamps can be joint and used together to prevent the DOS attack on the schemes.

Written passwords represent a significant security risk. Therefore, passwords must be memorized and saved in a safe way. As the size of passwords increases, remembering alphanumeric data of arbitrary sequences with a combination of special characters become tedious. A limited character set is feasible in real-world password options as the user must keep password management in mind [24]. In doing so, possible combinations to form passwords is effectively reduced. Choices for passwords drift in the direction of a dictionary and commonly used words. Thus, the entropy of a usable password is cumulatively reduced considerably. Hence, dictionary attacks, brute force attacks and guessing attacks are very easy to launch on these weak passwords [25].

Wang *et al.* [26] addressed the skewed nature of passwords by using 14 datasets that contained almost 113 million sample passwords. Their study outlines the biased distribution of commonly employed passwords. They also provide a method to establish the security strength of the used datasets. "PDF-Zipf" and "CDF-Zipf" are the two models that are proposed in their work. Natural language processing technique [27] is used by them to propose distribution models for the used datasets. To calculate the entropy of password dataset, a new metric based on their findings has also been proposed. Successful guesses during the attack and dataset sizes are the main factors

which influence the metric. [28, 29]. For this study, an English dataset consisting of “Flirtlife.de” and “RockYou” password dictionaries are used. With Unicode [30] taking over the internet, passwords in other more obscure languages such as “أرذو” are becoming more and more prevalent. To cater to these emerging circumstances, newer models that can be generalized to support all languages are required.

### 2.2.2 Key-based Authentication

Key-based authentication techniques [31, 32] are a type of techniques which are based on something you know. As compared to passwords which require an authenticatee (the entity to be authenticated) to transfer authentication data, a solution to a complex mathematical problem is required in key-based authentication schemes for authentication [33]. In key-based authentication, no authentication information or keys that are used as a substitute for passwords are communicated. These schemes offer a higher security level by securing the authentication information against eavesdropping attacks. Public keys stored on server-side databases offer the benefit of being resistant to theft even if the server is compromised [34]. Attacks such as database enumeration [35] will only reveal public information, *i.e.* public keys.

In [36], a unique authentication scheme is developed by employing public key cryptography formed on integer factorization problem and generalized discrete logarithm problem. They have compared their scheme with some other related schemes and proved their scheme is vastly superior as compared to self-certified [37] authentication schemes.

Key-based authentication schemes are typically computationally exhaustive because public key cryptography requires more computational resources for processing and therefore, it becomes difficult for low-power, low-performance IoT devices to use public-key cryptography.

Sciancalepore *et al.* in [38] presented a unique scheme for low performance IoT devices that allowed them to efficiently establish a connection with lower computation cost efficiently. Their proposed key management protocol (KMP) easily integrates elliptic curve Diffie-Helman and implicit certificates. Relevant experimental results and proof of concept implementation proves that their proposed KMP ensures maximum transmission time saving (up to 86.6%) with respect to traditional methodologies, robust key negotiation, fast re-keying and efficient security against replay attacks.

### 2.2.3 Multi-factor Authentication

The problems encountered while employing conventional authentication methods that are based on the concept of something known, something possessed, or something owned by the entity that needs to be authenticated has been the primary reason for the development of the multi-factor authentication model [39]. The multi-factor authentication paradigm integrates multiple authentication factors into a single authentication scheme thereby increasing complexity.

Alizai *et al.* [33] presented an authentication scheme that incorporates multiple factors to achieve IoT device security. Their scheme uses digital signatures and the device's unique ability to efficiently authenticate devices.

Multi-factor approach coupled with device capability has uniquely enhanced their technique to be resistant to attack vectors such as man-in-the-middle and replay attacks. However, the flaw in their scheme is that they are using standard public-key cryptography which is often not suitable for IoT devices because such devices cannot perform complex calculations efficiently.

Multi-factor authentication offers more security when compared with conventional authentication techniques [40]. However, new attack vectors and an increase in attack surface is also amplified due to the inclusion of multi-factor authentication techniques. Side-channel [41] vulnerabilities can be exploited to attack multi-factor authentication techniques as multi-factor authentication techniques often rely on side-channel communication mediums. For example, with higher signal strength software-defined base station, authentication calls and authentication SMS can easily be spoofed. Authentication procedures that rely excessively on out of band channels and their integrity can be easily attacked using these attack vectors [42].

## 2.2.4 Biometric Authentication

Everyone has unique characteristics; these characteristics can be used to distinctly identify any individual. These characteristics and traits are largely divided into two types of biometrics, namely, physiological and behavioral. Biometric features can include fingerprints, iris, retina, palm area, DNA, face, sounds, signatures or keystroke recognition.

Rathod *et al.* [43] have performed a detailed survey of fingerprint recognition systems. In their study, they have shown that fingerprints are the oldest



and most commonly used biometric recognition systems. False acceptance rate and false rejection rate of the biometric recognition systems, as well as loopholes and lacking areas of each scheme that they have analyzed, is also discussed.

To authenticate a person more securely and effectively, multiple biometric features [44] can be used instead of a single biometric feature. To authenticate as a legitimate user, the attacker will need to generate and distort all kinds of employed biometric information in scenarios where multiple biometric features are used. For example, it becomes a difficult task for an attacker to get an image of the iris and a fingerprint of enough quality at the same time; therefore, the attack will be difficult to execute. In [44], a multi-biometric framework is presented as well as various issues such as multi-data database, soft multi-biometrics, multi-algorithms fusion methods, identification of identical twins, indexing search and embedded hybrid recognition system are discussed which must be kept in mind at the time of designing a multi-biometric framework.

### **2.2.5 Hardware Authentication**

Another perspective that makes it possible to uniquely distinguish any entity is “something you have”. For authentication, items such as smart cards [45], RFID tags [46], USB token keys [47] can be used. Physical device attributes such as PUF [48] can be used to achieve hardware authentication. Characteristics of the equipment are used as a basis for PUFs. These characteristics are difficult to reproduce. These characteristics depend on many

factors like inherent characteristics introduced by fabrication; materials and environmental noise etc.

Much has been done for standardization of hardware-based authentication, but it is still an active research domain. Hardware security modules (HSMs) [49] are standard hardware-based implementations. Encryption capabilities are provided to other devices such as connected devices or the network by HSM as it relies on the presence of an integrated cryptographic co-processor. Purchasing price of HSMs is high. Manufacturers of the latest advanced computing devices have begun to integrate the Trusted Platform Module (TPM) [50] onto their devices. They are very similar to HSMs in their functions; however, they have a direct relationship with the device in which they were integrated at the time of production. Revocation of keys is also a major challenge if a TPM is compromised, as it is not feasible to simply change a device [51]. Major services that TPM provides are that it stores, generate and limit the use of cryptographic keys. TPM can also be used for platform device authentication using specific RSA key [52].

## 2.3 Session Management

There is generally a requirement for a technique that allows the authentication of succeeding requests after a successful authentication without having to re-authenticate. Managing consistent sessions is an important part of the web application. It links succeeding user requests and other stateful information together to give rich and interactive user experience. But the existing standard by which session is managed is cookie-based. It is not a perfect

way to manage a session as cookie-based session management is vulnerable to attacks like session hijacking and session fixation attacks. HTTP achieves a stateful behavior by employing technologies such as cookies and tokens [53] as it is stateless by default. Session management is performed by TLS using identifiers that are transmitted as a part of the initial request messages from the server [4].

In [11], a technique has been proposed for secure session management based on a shared secret between both parties, an incremental counter and HMAC is employed for managing sessions [54]; however, this technique introduces performance and network overheads.

In [3], the authors discussed existing threats against session management as well as currently deployed countermeasures to secure sessions, including HTTP only and secure cookie flags. Authors presented a mechanism called “SecSess” to secure a session. The proposed mechanism provides security to sessions and is compatible with common web caches. But the problem with “SecSess” is that it introduces performance and network overhead.

Session management in the application layer is not standardized as it is in transport layer protocols such as TLS. Security issues related to session management are among the top 10 issues of application security for the Open Web Application Security Project (OWASP) [10]. Session management mechanisms at the lower levels of the communication stack do not translate enough session information upwards to the application layer. Applications not maintaining stateful connections do not maintain session information even at the lower layers and rely on a one-time request and response pair [17]. Every request is treated as its own one-time session that is independent

of each previous interaction. Session management is usually custom created in each software solution by the developers. An objective of the presented work is to incorporate sessions established at lower layers such as the network layer into the application layer. Doing so results in combining independent sessions created at different layers of the communications stack into a single more robust session management mechanism. Such a mechanism provides better security and stronger cryptographic guarantees. This scheme results in the creation of a unified standard session management mechanism that can be incorporated into every software solution and does not require re-engineering with every implementation, resultantly reducing the time and complexity required for the implementation of session related security issues. The presented scheme can serve as a standard for all custom-built session management solutions.

## 2.4 Cookies and Tokens

The most widely accepted technique for authentication and continuous authentication is cookies and tokens. Protocols such as HTTP and HTTPS make use of cookies for maintaining a stateful session regardless of the stateless underlying architecture. Malicious cookies obtained as a result of cookie theft will breakdown any session management technique that relies on the integrity of cookies. The attacker has a wide range of techniques to simulate the session of the application, including session fixation [55], cookie theft [56] and token forgery [55]. An attacker who has no knowledge of the authentication credentials can regardless of his limitation, create fake sessions if he is in

possession of a stolen cookie [57]. A limited number of information systems rely on short lived cookies to cater for stolen cookies. Reducing the session timeouts, after which the user is required to re-enter their credentials is more of a usability hazard rather than being a defense against an adequately equipped attacker. An attacker with adequate knowledge will require a very short time to carry out the malicious tasks and get out of a compromised system.

A stateless environment requires system developers to implement the appropriate mechanisms [55]. Tokens used in the service requests present a solution to the cookie theft issue. Tokens resemble cookies in many aspects, however their dynamic nature and non-reliance on local storage provides an edge over cookies regarding security requirements of a system. A Token must be sent with every service request just like a cookie. Tokens are embedded into the application's procedural logic, for example, long random strings are inserted into the POST/GET requests, by attaching them to the end of the URLs for web resources. The use of a static token or the lack of randomization in a token can lead to session fixation attacks [58]. Using low entropy seeds and improper randomization can also result in session fixation attacks. For service providers, serving thousands of requests per second, generating thousands of random numbers is an intensive task. Generating random numbers that can be effectively used in conjunction with the token generation process while preserving adequate entropy is an expensive task. Invalidating the token requires maintaining extensive blacklist and user must search for and access this blacklist for each request being processed.

An in-depth study was carried out on the privacy and relevance issues of

cookies used by the top ranked (Alexa Ranking) hundred thousand websites [59]. The study reveals significant occurrences of insecure cookies that can be easily attacked. X. Zheng *et al.* [60] studied important websites such as google.com and Bank of America and found out important cookie related vulnerabilities in these websites. In addition, the study also analyzed how these vulnerabilities adversely affected these websites, along with the implications of weak cookie implementations in widely used browsers.

## 2.5 Transport layer Security Mechanisms

Data protection while it is in motion is of paramount importance for any information system's secure functioning. Communication security is commonly achieved by protocols such as TLS. TLS not only provides data confidentiality, but tamper proofs the data. Any adversary controlling the network between the sender and the receiver cannot eavesdrop the communication. TLS can work with a multitude of key agreement protocols; it also provides support for different cryptographic guarantees based on the type of cipher suite used [4]. Each cipher suite provisions a specific set of security primitives and safeguards.

TLS focuses primarily on establishing a secure and encrypted communication path between the sender and the receiver [4]. A study by Dietz *et al.* [61] presented a technique primarily based on public-key cryptography. Their technique achieves clients side authentication over the internet by extending TLS using an extension known as TLS Origin Bound Certificates (TLS-OBC). Clients can create authenticated channels with their

servers using this TLS extension. They have used TLS-OBC to link authentication tokens that already exist, such as HTTP cookies to authentication channels. Their technique enhances the authentication security using OBC. Their technique is compatible with existing network infrastructure. It is hard in general to distinguish between service by examining the TLS traffic. Kim *et al.* [62] proposed a new method for creating service signatures using data payloads extracted from TLS packets. They have achieved 90% efficiency in the classification of encrypted TLS traffic based on the source service or source application.

## 2.6 Summary

This chapter covers the background and the related work of the thesis. The related literature has been presented along with a critical analysis of the studies. Previous research work and schemes used in the literature helps in formulating the solution to the identified problem. In the following chapter, we will discuss the research methodology that has been followed during the thesis.

# Chapter 3

## Research Methodology

Chapter 3 explains the research methodology that is followed to carry out this thesis research. For the presented thesis, hybrid approach is selected after the complete analysis of present research methods. As every research method is appropriate for different research scenarios of this thesis, hence all these research methods are followed at different times. A brief description of the methods that are used in our research methodology along with the phases followed in the research process, *i.e.* identifying the problem, formulating hypothesis, making important observations and evaluating the system is given in this chapter.

### 3.1 Introduction

The research is the procedure of probing or collecting information specific to the area under consideration. It can be defined as a scientific investigation to discover new information and facts [63]. Whenever there is a need to answer



a problem and finding appropriate solution to that problem, research is performed [64]. As stated by Clifford Woody, research is the process of defining and redefining the known problem, formulating hypothesis and suggesting solutions to the known problems, assessing the collected data, making assumptions, deriving conclusion and lastly testing the conclusion for verifying the stated hypothesis [65].

### 3.1.1 Research Types

Research methodologies which were proposed in the literature are given below and described in figure 3.1.

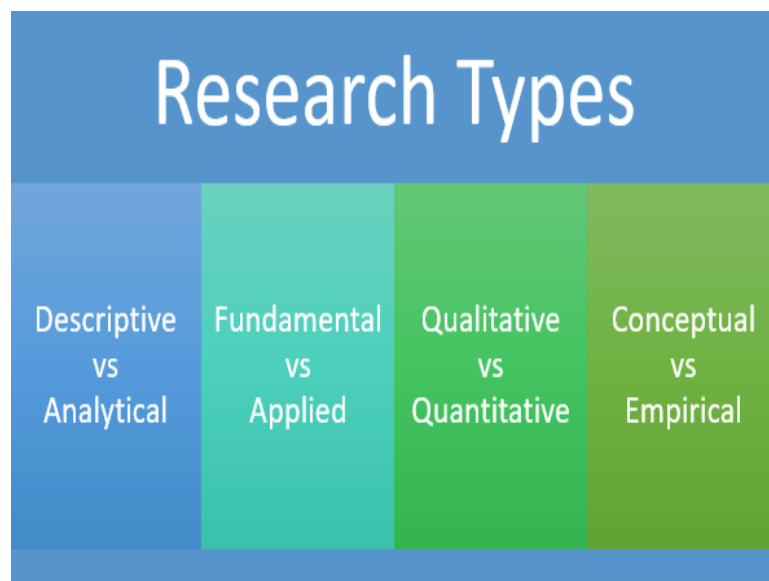


Figure 3.1: Research Types

**Descriptive vs Analytical Research:** In descriptive research, the present work is discovered, and small reviews are conducted to find the essential and linked information and proofs. Its objective is to define the work about the

specific area of research which can additionally be used in upcoming research work. The main features of descriptive research are that the researcher has no control over the data and present literature; rather they can only find the proofs and methods. In analytical research, the researcher uses all the gathered information from the reviews and do the critical analysis and assessment to reach the conclusion [63, 64].

**Fundamental vs Applied Research:** The research can either be fundamental or applied depending on the depth of the knowledge required. In fundamental research, the main purpose is to collect basic knowledge and discover the basics of the scientific phenomenon. In applied research, the focus of the researcher is to find some suitable solution to the problem faced by other researchers, organizations or society. Different experiments are performed by the researcher to investigate and analyze the problem and achieve in-depth knowledge of the problem area. The approach to find solutions of different problems on the based on performed experiments is very helpful [63, 64].

**Quantitative vs Qualitative Research:** Quantitative research is based on the quantitative measurement of some features. It can be done in domains where things can be stated with respect to quantity. Whereas in qualitative research, things assessed based on the quality, e.g. the human conduct or opinions on specific things and the purpose behind those opinions [63].

**Conceptual vs Empirical Research:** Conceptual research is based on abstract thoughts or the concepts normally used by the theorists. This is

done to create new concepts or re-design the present ideas. The empirical research is based on the observations and experiments without trusting on any concept or the scheme. It is truly based on their own set of observations, experiments, and conclusions. In this type of research, firstly, hypothesis is made based on the facts and then try to assume the outcomes. Researcher then collects the proofs to accept or reject the hypothesis [63].

### **3.1.2 Research Methods and Research Methodology Overview**

To conduct the research, different techniques, methods and procedures are used by the researchers which is called research methodology. The process is started with conducting reviews until reaching the results and this as a whole is termed as research methods. The research methodology is defined as the principles and procedures to carry out research using a scientific approach. It includes all the stages used by the researcher to find a solution to the problem. The research methodology is different for different types of research problems, and the researcher must know the correct way to be used to follow the research. The research objective should be very clear to the researcher towards the selection of a research methodology [63, 64]. In this thesis following research methodology steps are used:

- Explore the session management, authentication, continuous authentication, Transport layer security mechanisms (TLS), Application Layer security, attacks and challenges to gather significant knowledge for the targeted domain.

- Narrow down the study to choose some challenges to work on *i.e.* cookie theft and token forgery attacks that happens at application layer even in the presence of TLS.
- Derive hypothesis from the analysis of the existing literature.
- Validate the formulated hypothesis after implementation of the proposed system.

## 3.2 Thesis Research Methodology

This thesis used the hybrid approach throughout the research, which includes conceptual, applied and fundamental research methods. This thesis consists of some steps which are followed throughout the research phases starting from collecting information till obtaining the results which are then assessed. All the steps in our research process are given below.

- Define a Research Area
- Literature Survey
- Identify Research Problem
- Develop Hypothesis
- Observations
- Prototype Implementation
- Hypothesis Testing

### **3.3 Summary**

In this chapter we have covered different research methodologies that have been proposed for research which can be followed by the researchers. As every research method is appropriate for different research scenarios of this thesis, hence all these research methods are followed at different times. In next chapter we will look at the proposed solution to the problem that has been identified in the previous chapters.

# Chapter 4

## Proposed Solution

Chapter 4 explains the proposed framework that has been designed to mitigate different attack vectors. This chapter also discusses the primitives that are used in the framework and are essential for understanding the correct working of the framework. Security provisions of the framework are also discussed in this chapter.

### 4.1 Scheme Primitives

The primitives used in the proposed scheme are discussed one by one below. Knowledge of these primitives is helpful for understanding the proposed scheme.

#### 4.1.1 Timestamps

Timestamps are very important to keep track of data when it is moved, deleted or created online. A timestamp is the present time of an incident

that is recorded by a system. A timestamp is used to avoid replay attacks. For example, a timestamp is used in Kerberos to prevent replay attacks. Timestamp has been used in the proposed scheme for similar purposes.

### 4.1.2 Random Numbers

Random numbers are needed in cryptography to generate cryptographic keys, also used as a nonce *i.e.* to verify and timestamp, random numbers are also used as salts which are used as arbitrary inputs in hash functions, and one-time pads. Random numbers are utilized to make deterministic patterns unclear. They are also used to remove human biases from pad messages by increasing the length of the messages. There are two types of random numbers, true random (TRNG) and pseudo-random (PRNG). TRNG's are the type of random numbers in which an unpredictable physical means of the system is used to generate a number. Whereas in PRNG's, mathematical operations and algorithms are used to generate a random number.

### 4.1.3 Symmetric Cryptography

Symmetric key cryptography uses a single key for both encrypting as well as decrypting the text. Therefore, it is frequently called same, shared or single key encryption. Since the key is kept secret, it is also called private or secret key encryption. Symmetric algorithms are simple, fast and are cost effective in terms of efficiency. The downside of symmetric key cryptography is key-exchange, too many keys, and since both sender and receiver use the same key, messages cannot be verified to have come from a user. The exposure of

the key is undetected when using symmetric key.

#### 4.1.4 Asymmetric Cryptography

Asymmetric cryptography also known as public key cryptography is used to overcome the downsides and limitations of the symmetric key cryptography. While at first maintenance of the key is required, the basics of public key cryptography give an elastic and extensible structure in which the cryptographic functions can be deployed. Asymmetric cryptography provides message authentication, non-repudiation, integrity, easy key distributions. The downside of asymmetric cryptography is it is slow and require computational resources. The user of the public key should ensure that public key belongs to the intended owner.

#### 4.1.5 Hash Functions

A hash also known as message digest is a representation of large message in a smaller fixed length. Hashes are also known as one-way function *i.e.* the original message cannot be recovered from a digest. To ensure integrity and authenticity of the information, hashes are used. Even one bit of change in original message changes the hash considerably due to the avalanche effect. Hashes are not used to provide confidentiality. There are different hashing algorithms like MD4, MD5, SHA1, SHA2, SHA3 that takes the input of arbitrary length and gives output of fix length.



### 4.1.6 Message Authentication Code (MAC)

MAC is a type of data which is created to verify the authenticity of the message. It is created by appending the secret key with the message and then taking the hash of the whole block. At the receiving end, the user will use his secret key to create a MAC block and if the received and created MAC block are same then it will imply that the message has not been changed intentionally or accidentally in transit.

## 4.2 Scheme Features

Following is the list of functionalities that are given by the proposed scheme.

- Mutual authentication to two or more than two parties is provided by the proposed scheme using a certificate that is exchanged at the time of registration.
- User authentication forms the basis for creation of sessions.
- For each session, cryptographic keys are derived and then discarded, thus keys are never stored on the device.
- Continuous authentication is provided by the scheme based on a session key derived for specific time.
- All communication is symmetrically encrypted and integrity checks (MAC) are applied thus eliminating the need for cookies and tokens. Ultimately resulting in secure sessions that are not vulnerable to attacks like cookie theft and token forgery.

- Session timeouts are enforced because the derived session key is usable for a specific time.
- Access control mechanisms can be coupled with session identifiers to provide better access control for the resources.
- Communication can be made secure because of derived keys from session and authentication information.
- The scheme can be extended to provision a secure group communication scenario in a dictated key distribution fashion.

### 4.3 Proposed Solution

This research proposes a scheme for achieving authentication, continuous authentication, along with session management at the application layer. This scheme exploits the fact that a user registration process exists for every service that needs to be accessed by users. A side channel can be efficiently used for public key exchanges. Figure 4.1 shows that side channels can be used to carry out public key exchanges. These channels such as e-mail, are typically used for multi-factor authentication in a traditional setup. Table 4.1 explains the notations that are used throughout the scheme.

The key exchange over a side channel ensures that these public keys can be later used to establish a secure session without using a cookie or a token. After the key exchange, session establishment may occur at any time. This process is shown as a sequence of steps in figure 4.1. The steps illustrated in the figure are discussed in the following sections.

Table 4.1: Table of Notations

Symbol	Notation
$S$	Server's Identity
$C$	Client's Identity
$R_C$	Client Random no.
$R_S$	Server Random no.
$Enc_{PUB}$	Encryption with Public Key
$Dec_{PK}$	Decryption with Private Key
$  $	Concatenation
$S_K$	Session Key

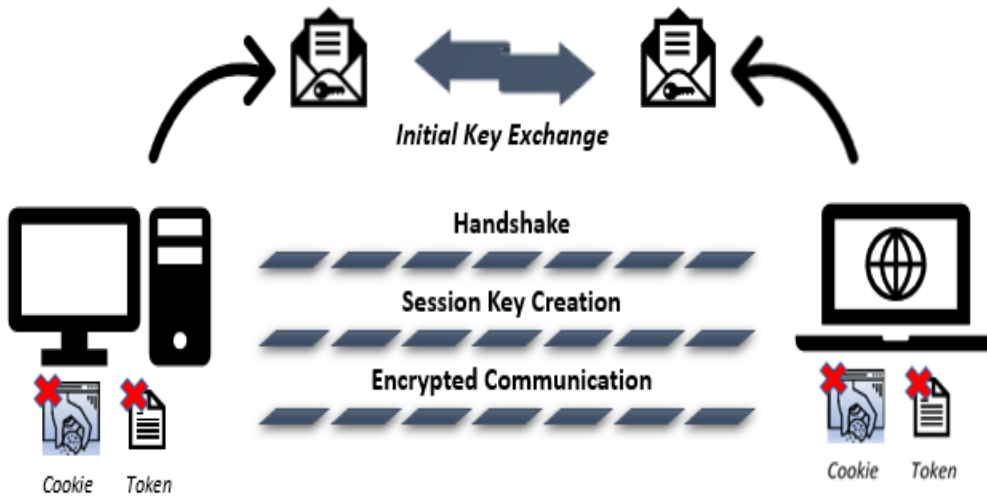


Figure 4.1: Proposed Scheme

### 4.3.1 Registration Phase

Traditionally account setup prerequisites include a registration sequence. The registration phase is used as a precursor allowing user to access the service. The registration phase is used for creating authorization data, presenting authentication provides access. However, the proposed scheme extends the functionality of the registration phase by using it for the mutual authentication of both the service requisitioner, *i.e.* the client and the ser-

vice provider, *i.e.* the server. Registration processes rely on the usage of out of band channels like e-mail, GSM etc. Thus, allowing the association of a public key to the digital identity that is being presented at the registration time. Sample certificate is shown in figure 4.2.

```
"Email": "samplemail@samplemail.com",  
"Phone no": "920001112224",  
"User's Public Key":  
"1515PralkedbNf0TpOG6M1DyR4e97042ZwIDA  
QA8Ao6AFjko56+0yM8M0RVyaRAXx++xTcP8Lh  
\n3tx4VgMtrp+11E8CjhoTwo23KMBALOGSYnRi  
so8ZM31MfTKevlkAidPEXvYCD05dYq3noLkkLy5L  
2\nplIVOFMDG+KESnAFV712c+cnzW4404.b6f8  
mR1C1z2oxyLL6002fuLi55L2\nplIVOFMDG+KES  
nAFV712c+cnzW4404.b6f8mR1C1z2oxyLL6002f  
uLi55/abSYxECQQDeAw6fiIQX\n",  
"Issue Date": "21/01/2018",  
"Expires On": "21/01/2019"
```

Figure 4.2: Sample Certificate

The server receives the client's public key, along with his registration credentials (email, phone no etc.). The role of CA is not important here because the server can very easily verify its authenticity using the side channel that was previously used for multi-factor authentication, e.g. an email with verification links etc. Similarly, the client will receive the server's public key using the exact same side channel that was used before, for example an email. The service provider's identity will be bound to its public key for all future use, however, to verify the actual identity of the service provider and to relate it with their domain information, a certification authority can intervene.

### 4.3.2 Authentication Phase

The server and the client can mutually authenticate each other if both parties have access to each other's public key. The authenticator *i.e.* the server initiates a challenge response scheme every time it receives a request containing identity information (ID tag) from the authenticatee, *i.e.* the client. In the first step, random number  $R_S$ , server's identification  $S$  and a timestamp  $T$  are sent to the client. This message is encrypted with the client's public key. Upon receiving the challenge, the client will decrypt the received challenge and generate its own random number. Then with the received challenge, the client will concatenate its generated number  $R_C$  and its identification  $C$ . The client will calculate its hash and send the hash and a random number  $R_C$  encrypted with public key of the server back to the server. The server, upon receiving the challenge, will decrypt the received identity  $C$ , the random number  $R_C$ . After decryption, the  $R_C$  is concatenated with the previously sent challenge. This newly created string is again hashed and used for verification. If the new string's hash and the received hash are both equal, then the client is correctly authenticated. The process is shown in the figure 4.3 that after the establishment of digital identities along with public keys, it is trivial to establish mutual authentication for both client and server.

### 4.3.3 Session Key Establishment

Once the client and the server possess each other's randomly generated numbers, they can create symmetric keys based on these numbers. Any future communication will be encrypted using that symmetric key. Hence there will

be no future requirement for re-initiating a challenge response session. To obtain a symmetric session key, each party must simply concatenate both random numbers  $R_C$  and  $R_S$ . After concatenating, a hash is taken using a hash function like SHA-256 [66]. The hash value is then attached to the client's ID  $C$  which is previously stored on the server. This hash value can be used as the key for session encryption when used in conjunction with the AES 256-bit symmetric encryption algorithm. This key will remain active until the session ends or expires. After the establishment of the session key  $S_K$ , both communicating parties can easily verify each incoming message's origin. The usage of the same encryption key that was computed during the key exchange phase guarantees that the message has been sent by an authenticated party. Appending message authentication code (MAC) [67] ensures message's integrity whereas an appended identification (ID) tag which contains the identity information of the client enables source authentication.

#### 4.3.4 Scheme Handshake

As shown in figure 4.3, an authentication request is sent in the first step by the client. As a response to the request, the server presents a challenge to the client as step 2. Decryption of the encrypted challenge enables the client to successfully complete the challenge and end the negotiation process. Being able to decrypt the challenge also serves as the proof of identity for the client. The decrypted data that was obtained during step three is used in the making of the session key  $S_K$ . Knowledge of the freshly created session key is an implicit proof of knowing the correct decryption data in step two,

which in turn proves the authenticity of a client claim. At the server side, similar events lead to the server being authenticated. As seen in step four, the server is sent an encrypted challenge. If the server is capable of correctly decrypting the challenge, it will obtain all the necessary data which will enable it to create a session key  $S_K$ . Knowing the correct symmetric key will imply the authenticity of the server. Therefore, the server and the client are both mutually authenticated.

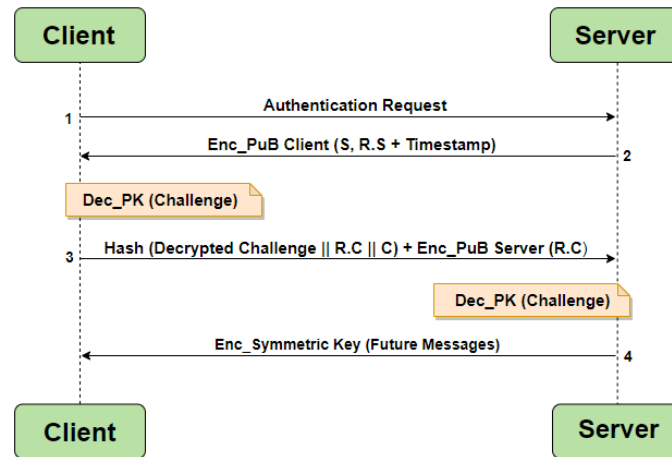


Figure 4.3: Scheme Handshake

## 4.4 Security Provisions

Every message is validated, and it has achieved authentication and continuous authentication. World wide web is a complex environment and session management in these environments primarily depends upon tokens and cookies. If a message is successfully validated, this indicates that the message is part of the session, thereby eliminating the need for cookies and tokens. Access control mechanisms can be coupled with sessions for the provision of

different resources.

The scheme uses cookie-less and token-less methodologies for authentication, mutual authentication, and continuous authentication based on cryptographically secure primitives like timestamps (to avoid replay attacks), random numbers (for key generation and incorporate randomness), asymmetric cryptography (for non-repudiation), symmetric key (for secure communication) and MAC (for message integrity). Hence mitigating attacks such as cookie theft and token forgery. Session management and authentication is achieved by the proposed scheme using asymmetric key cryptography.

## 4.5 Summary

In this chapter we have discussed the proposed solution for the identified problem. The different steps used in the framework are also explained in detail. This chapter also discusses the outcomes and security provisions that the proposed scheme will provide. In the next chapter we will look at the implementation and test results of the presented framework.



# Chapter 5

## Implementation and Results

Chapter 5 of the discusses the implementation of the proposed solution and the technical details. This chapter also discusses the results obtained from two different environments. The chapter also includes a hypothetical case study that helps in understanding the scheme as well as its security aspects.

### 5.1 Scheme Analysis

The presented scheme is analyzed using a network protocol analyzing tool called Scyther [68]. The tool works by using a script that follows a specific rule set. Scyther is a tool that verifies a cryptographic protocol against different type of attacks such as secrecy, replay, man-in-the-middle and aliveness attacks.

The outcome of the analysis is presented in figure 5.6. For analyzing the security of the scheme, a Network Threat Model [69] is simulated to verify the scheme. It is assumed that:

- The network is fully or partially in control of the attacker.
- As discussed in Dolev-Yao intruder model [66], the attacker is very resourceful and can learn, create and deflect messages.

## 5.2 Attributes

Scyther's attributes like synchronization, aliveness, protection against man-in-the-middle and secrecy that are considered when verifying a protocol. These attributes are discussed in detail below.

### 5.2.1 Secrecy

First guarantee that is provided by the presented scheme is that it provides confidentiality to user's credentials. Analysis of the presented scheme shows that on untrusted network the communication between both parties will not be uncovered by any adversary. As shown in figure 5.6, random numbers of the client, the server and the timestamp utilized in the exchange stayed undisclosed.

### 5.2.2 Aliveness

Second guarantee that is provided by the presented scheme is the aliveness property. The aliveness property guarantees that the response from the responding party is the result of the request that was generated by the communicating party. It likewise guarantees that the communication between both parties is not altered and messages are correctly timestamped and digitally

signed.

### 5.2.3 Synchronization

Third guarantee of the presented scheme is that it addresses the Non-injective Synchronization (Nisynch) property [70]. Thus, the scheme provides protection against replay attacks. Nisynch property guarantees that communication between receiving and sending party is synchronized and is initiated by the sending party.

### 5.2.4 Man-In-The-Middle

For all the tests that we conducted, no attack tree was produced. Hence, we can say that the presented scheme gives protection against man-in-the-middle attacks.

## 5.3 Scyther Script

Properties of Scyther *i.e.* synchronization, aliveness, protection against man-in-the-middle and secrecy are tested and validated using the Scyther script that is given one by one in the figures below. Client and the server are tested and validated individually.

To declare a new variable, keyword “fresh” is used. Timestamp and nonce creation are shown in figure 5.1. The variable Nonce, ServerRandom and ServerData are initialized whereas timestamp is initialized for ServerTimestamp in server role. Similarly, in client role, Nonce variable is initialized for ClientRandom and ClientData.

```
fresh VariableName1: Nonce
fresh VariableName2: Timestamp
```

Figure 5.1: Scyther Script - 1

To create a new variable that stores the values of “fresh” type variable, keyword “var” is used. In Scyther, the way new variables are created to store the values of the received terms is shown in figure 5.2. ServerRandom, ServerData, ServerTimestamp are initialized as variables to store the values received from server in the client role. Similarly, ClientRandom, ClientData are initialized as variables to store the values received from client in the server role.

```
var VariableName1: Nonce
var VariableName2: Timestamp
```

Figure 5.2: Scyther Script - 2

Communication between communicating parties is simulated with two function namely send\_1 and rcv\_1. The script works in a way that when a send function is called, there should be an agreeing function at the receiving end. These function calls must be appended with the same number to work correctly. The figure 5.3 shows how a message is sent from one party to another.

```
send_1(
  SenderAgent,ReceivingAgent,
  {SenderID,Message}pk(ReceivingAgent)
);

rcv_1(
  SenderAgent,ReceivingAgent,
  {SenderID,Message}pk(ReceivingAgent)
);
```

Figure 5.3: Scyther Script - 3

The figure 5.4 shows how a message encrypted with same keys is sent from one party to another. The communication is being encoded using the message hash  $H()$  of a key string that is generated during a protocol handshake.

```

send_2(
  SenderAgent,ReceivingAgent,
  {SenderID,Message}H(KeyString)
);

recv_2(
  SenderAgent,ReceivingAgent,
  {SenderID,Message}H(KeyString)
);

```

Figure 5.4: Scyther Script - 4

Figure 5.5 shows the properties that were claimed to hold during the protocol negotiation. These properties are analyzed and verified by the Scyther Protocol Verification Tool.

```

claim_Agent1(Agent,Alive)
claim_Agent2(Agent,Niagree)
claim_Agent3(Agent,Nisynch)
claim_Agent4(Agent,Weakagree)
claim_Agent7(Agent,Secret,Timestamp)
claim_Agent5(Agent,Secret,ClientData)
claim_Agent6(Agent,Secret,ServerData)
claim_Agent8(Agent,Secret,ClientRandom)
claim_Agent9(Agent,Secret,ServerRandom)

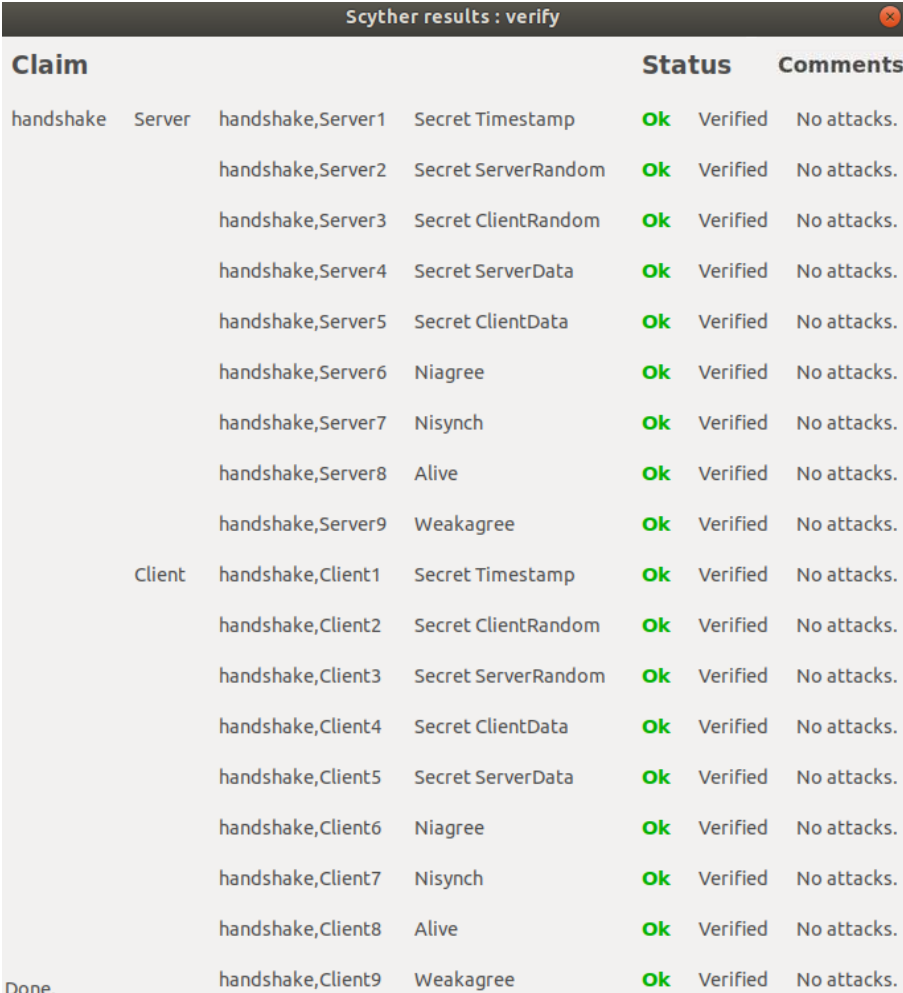
```

Figure 5.5: Scyther Script - 5

## 5.4 Scyther Results

The figure 5.6 shows Scyther generated results. Scyther verifies and validates properties of Scyther *i.e.* synchronization, aliveness, protection against

man-in-the-middle for both sides who are communicating. In this case the communicating parties are the server and the client.



Scyther results : verify				Status	Comments
Claim					
handshake	Server	handshake,Server1	Secret Timestamp	Ok	Verified No attacks.
		handshake,Server2	Secret ServerRandom	Ok	Verified No attacks.
		handshake,Server3	Secret ClientRandom	Ok	Verified No attacks.
		handshake,Server4	Secret ServerData	Ok	Verified No attacks.
		handshake,Server5	Secret ClientData	Ok	Verified No attacks.
		handshake,Server6	Niagree	Ok	Verified No attacks.
		handshake,Server7	Nisynch	Ok	Verified No attacks.
		handshake,Server8	Alive	Ok	Verified No attacks.
		handshake,Server9	Weakagree	Ok	Verified No attacks.
handshake	Client	handshake,Client1	Secret Timestamp	Ok	Verified No attacks.
		handshake,Client2	Secret ClientRandom	Ok	Verified No attacks.
		handshake,Client3	Secret ServerRandom	Ok	Verified No attacks.
		handshake,Client4	Secret ClientData	Ok	Verified No attacks.
		handshake,Client5	Secret ServerData	Ok	Verified No attacks.
		handshake,Client6	Niagree	Ok	Verified No attacks.
		handshake,Client7	Nisynch	Ok	Verified No attacks.
		handshake,Client8	Alive	Ok	Verified No attacks.
		handshake,Client9	Weakagree	Ok	Verified No attacks.
Done.					

Figure 5.6: Scyther Result

## 5.5 Testing and Implementation

This section throws light on the testing and implementation of the proposed scheme. The tools and specifications of the system on which the scheme is

implemented and tested are given in the following table 5.1.

Table 5.1: Tools & System Specifications

Name	Type/Version
Microsoft Windows 10	64bit
Ubuntu	19.04
Kali Linux	2018.4
Java	8
XAMPP	7.3.2
Php	7.3.2
JavaScript	—
Wireshark	2.6.6
Scyther	1.1.3
Burp Suite	1.7.36

### 5.5.1 Environment Setup

The proposed scheme has been tested in two distinct environments. *i.e.*

- A server client environment setup in java.
- A PHP script running on the server side and the client's browser running JavaScript.

The client and server were setup and the public key certificates were exchanged using TLS secure web forms emulating side channels. The same can be done by using minimal certificates (containing only the corresponding public keys) and placing them in the corresponding system's manually. An echo server was used as the service provider server. A service request was initiated by the client-side application. The request was answered by the service provider with an authentication challenge specifically crafted for the

authenticating client. After successfully completing a handshake, both the server and the client agreed upon a symmetric key and established a session as shown in figure 5.7.

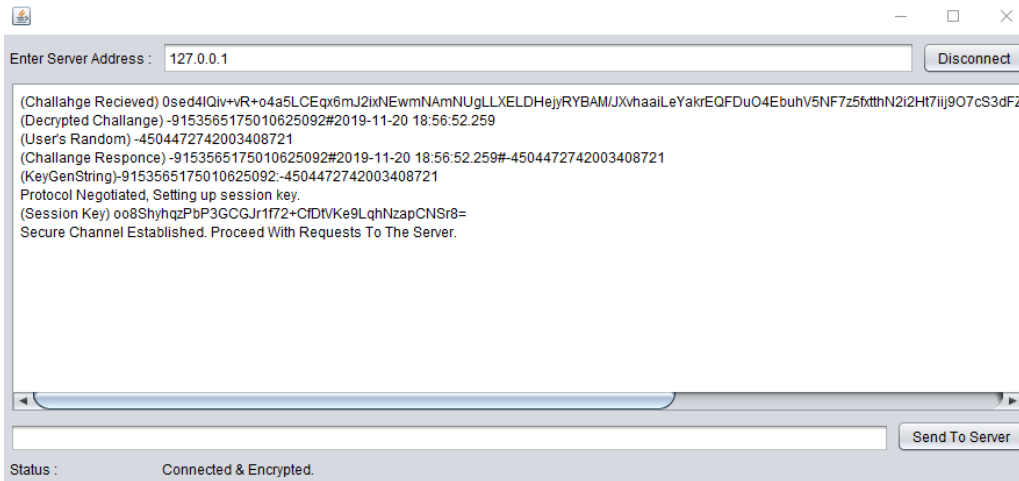


Figure 5.7: Client Server Application (JAVA)

Time	Source	Destination	Protocol	Length	Info
48.55.714332	192.168.43.223	192.168.43.244	TCP	66	50448 → 4567 [SYN] Seq=0 Win=17520 Len=0 MSS=1460 WS=256 SACK_PERM=1
48.55.714590	192.168.43.244	192.168.43.223	TCP	66	4567 → 50448 [SYN, ACK] Seq=0 Ack=1 Win=5535 Len=0 MSS=1460 WS=256 SACK_PERM=1
50.55.837439	192.168.43.223	192.168.43.244	TCP	54	50448 → 4567 [ACK] Seq=1 Ack=1 Win=17408 Len=0
52.56.871746	192.168.43.223	192.168.43.244	TCP	60	50448 → 4567 [PSH, ACK] Seq=1 Ack=1 Win=17408 Len=6
53.56.913266	192.168.43.244	192.168.43.223	TCP	54	4567 → 50448 [ACK] Seq=1 Ack=7 Win=131328 Len=0
54.56.942755	192.168.43.223	192.168.43.244	TCP	56	50448 → 4567 [PSH, ACK] Seq=7 Ack=1 Win=17408 Len=2
55.56.991488	192.168.43.244	192.168.43.223	TCP	54	4567 → 50448 [ACK] Seq=1 Ack=9 Win=131328 Len=0
56.59.229744	192.168.43.244	192.168.43.223	TCP	226	4567 → 50448 [PSH, ACK] Seq=1 Ack=9 Win=131328 Len=172
57.59.448085	192.168.43.223	192.168.43.244	TCP	54	50448 → 4567 [ACK] Seq=9 Ack=173 Win=17152 Len=0
58.59.448995	192.168.43.244	192.168.43.223	TCP	56	4567 → 50448 [PSH, ACK] Seq=173 Ack=9 Win=131328 Len=2
59.59.487405	192.168.43.223	192.168.43.244	TCP	54	50448 → 4567 [ACK] Seq=9 Ack=175 Win=17152 Len=0
60.59.668462	192.168.43.223	192.168.43.244	TCP	226	50448 → 4567 [PSH, ACK] Seq=9 Ack=175 Win=17152 Len=172
61.59.789973	192.168.43.244	192.168.43.223	TCP	54	4567 → 50448 [ACK] Seq=175 Ack=181 Win=131072 Len=0
62.59.715333	192.168.43.223	192.168.43.244	TCP	182	50448 → 4567 [PSH, ACK] Seq=181 Ack=175 Win=17152 Len=48
63.59.755398	192.168.43.244	192.168.43.223	TCP	59	4567 → 50448 [PSH, ACK] Seq=175 Ack=229 Win=131072 Len=5
64.59.756681	192.168.43.244	192.168.43.223	TCP	56	4567 → 50448 [FIN, PSH, ACK] Seq=180 Ack=229 Win=131072 Len=2

Frame 56: 226 bytes on wire (1808 bits), 226 bytes captured (1808 bits) on interface 0  
 > Ethernet II, Src: IntelCor\_22:ee:8f (fc:f8:aee:22:ee:8f), Dst: IntelCor\_Bc:27:88 (60:f6:77:8c:27:88)  
 > Internet Protocol Version 4, Src: 192.168.43.244, Dst: 192.168.43.223  
 > Transmission Control Protocol, Src Port: 4567, Dst Port: 50448, Seq: 1, Ack: 9, Len: 172  
 > Data (172 bytes)  
 Data: 4c6271345a52494157583069756f5a437a676f3157315a65...  
 Text: Lbq42R1A0W1u02Czgo1l12eW1wacsS1MdAFvMlxynxx+c4+P3pyI70BRLDKUm+FuWlCQfj1965CS0e0KHzgq6b2R152YU6Vyg/9jLQtw63ditExoi+/r4/0S3nr0/zu206699JLabfbcvcs//6e4FB/b20N7Zt7uxHP8g-[length: 172]

Figure 5.8: Traffic Capture - Wireshark

Cipher objects were instantiated using the symmetric key. The cipher objects corresponded to AES-CBC [71] and AES-GCM [72] during different trials. Server and client both were able to authenticate themselves, as well as keep their session active if the MAC verification was valid (corresponding



to the ID tag) *i.e.* if the MAC is valid, the authenticity of the message origin is guaranteed. Access control methodology was tested on top of the session. As shown in figure 5.8, data intercepted by Wireshark was analyzed and the communication data was found to be encrypted. The encryption was comparable to AES-CBC or AES-GCM modes of TLS.

This was also tested using a PHP script that was running on the server side and the client's browser was running JavaScript as shown in figure 5.9, figure 5.10, figure 5.11 and figure 5.12. Server and client information is highlighted in blue and red respectively whereas hashes taken at both sides are highlighted in green color.

```

localhost/appsecframework/ x +
localhost/appsecframework/
Challenge value received from server:
g2lJUcvou7fof71tagDCehnmxaA9N7H59zg5ft0grdrAFcgkyoUUElAJh0nTRsNsk23aN4zXxvErF8EI7a00MXMkCbPeNM/yGqZT9JcLUzKffUFT6Oxv+kTjdKrUsl//
Received challenge decrypted by client:
localhost:779b0f8c09c91f1327137b462cccdc4b:1574422001
Random number generated by client:
318468e7fc8791b681ae7b3b58539176
Proof Created by Client (Plain Text):
localhost:779b0f8c09c91f1327137b462cccdc4b:1574422001:318468e7fc8791b681ae7b3b58539176:email@email.email
Hash of Client's Proof:
1b03407ad2546600f54f3402546b6d1906660c978910728d55826acd49e34a08
Client's random encrypted with Server's Public Key
TKoxqByOAiGuY/KzAe/D4v1wzuHZii8No3nW7+TRpRxaIoXTUcvSBGv9Tiv5OiYzwRmf1OLk4k3iAzLVv28mLcfY76tBn6cizBfEAnFkgzpAz1WkRoSRJcNj5Cx

```

Figure 5.9: PHP - 1

The resulting scenario was very much comparable to the previously mentioned client server model. The same results were obtained as mentioned in the previous implementation. Symmetric session keys were successfully initialized, and the communication was encrypted using the established session

keys.

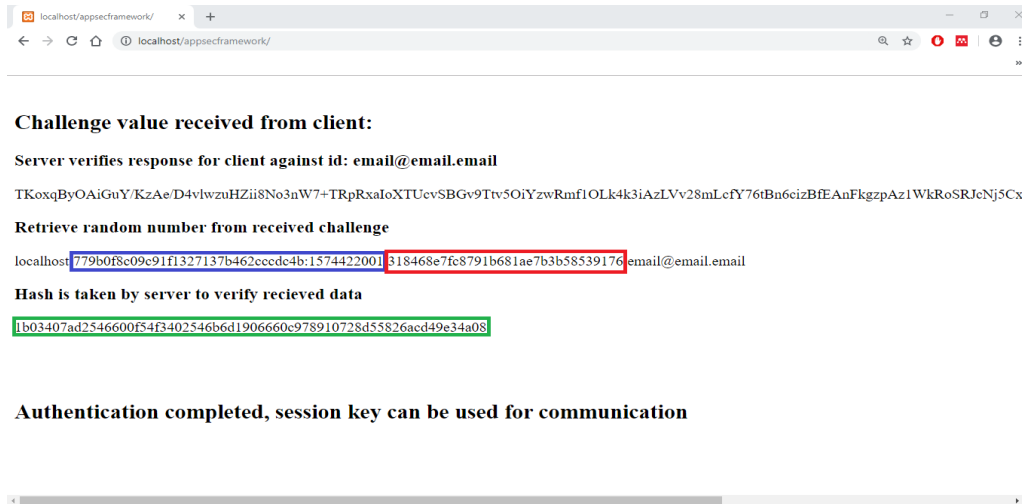


Figure 5.10: PHP - 2

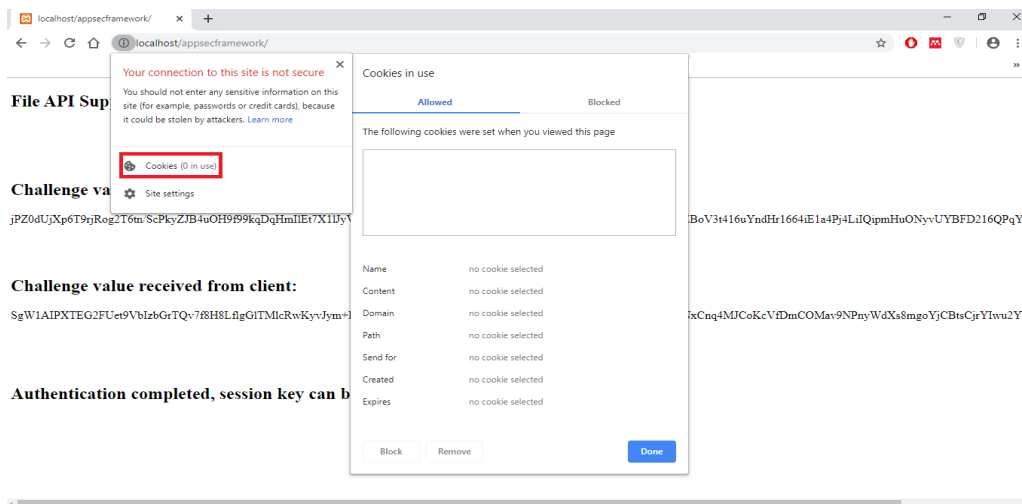


Figure 5.11: PHP - 3

The communication was tested using Burp Suite as shown in figure 5.13, figure 5.14, figure 5.15 and figure 5.16. Burp Suite was unable to generate any trace of the original information. The only visible information was the

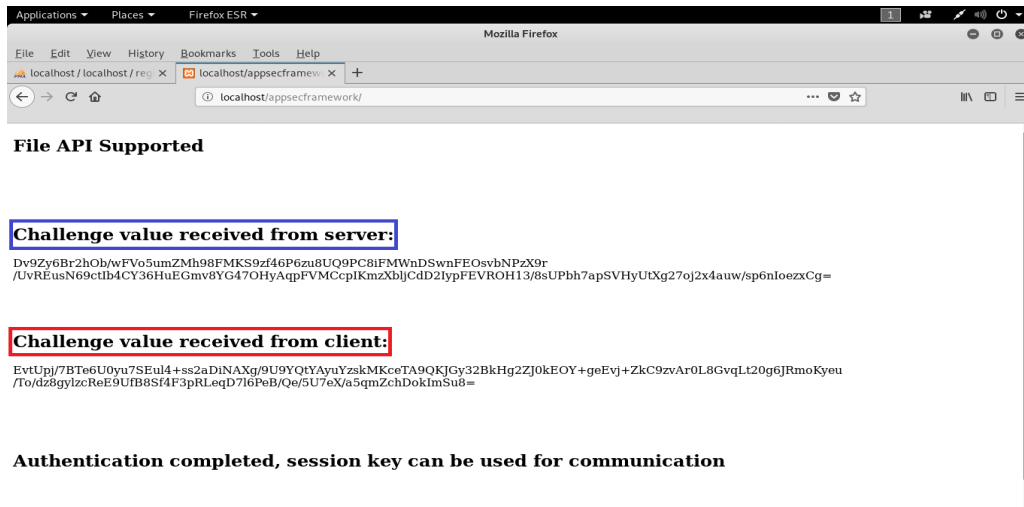


Figure 5.12: PHP - 4

unencrypted HTTP headers. The HTML data remained totally encrypted and immune to a man-in-the-middle attack. And the authentication was achieved without the use of session cookies as shown in figure 5.11 and figure 5.12.

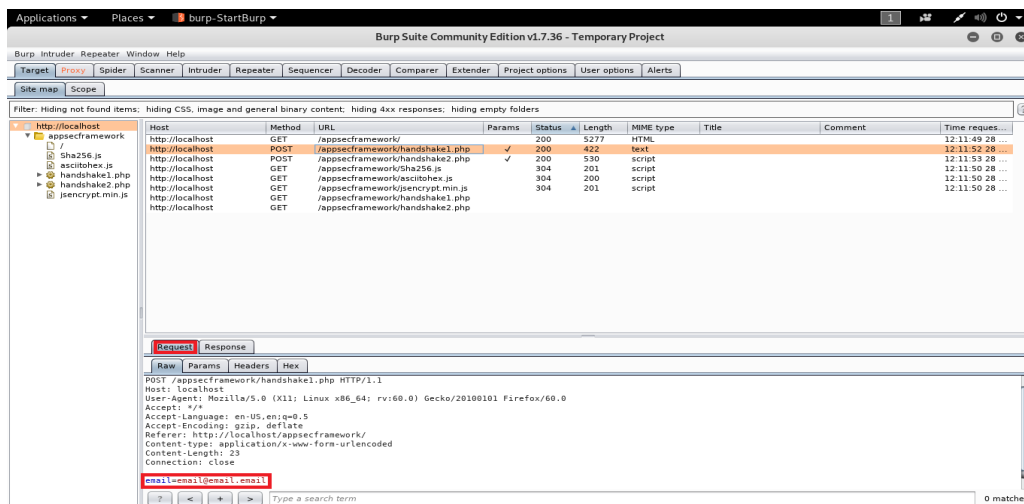


Figure 5.13: Traffic Capture - Burp Suite - 1

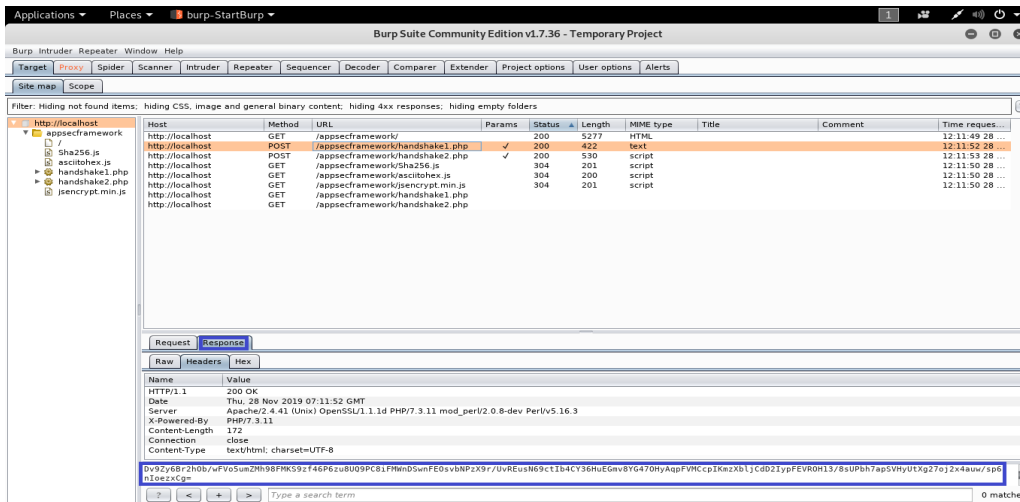


Figure 5.14: Traffic Capture - Burp Suite - 2

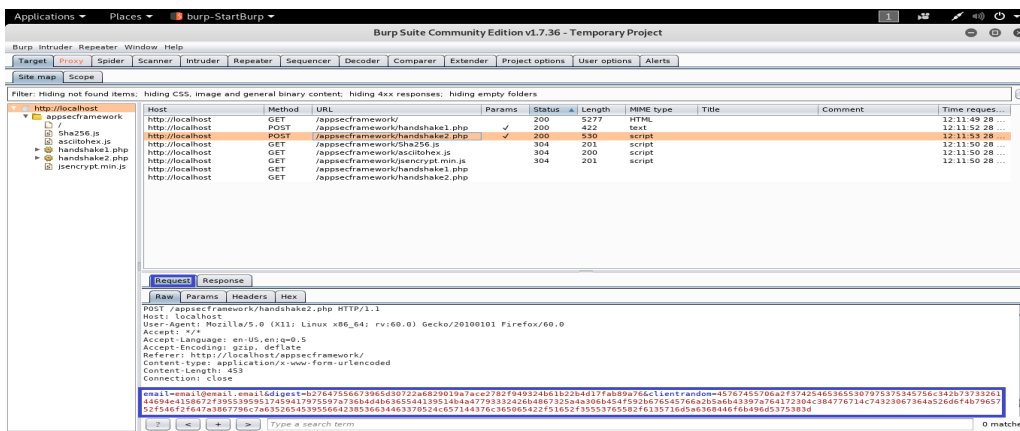


Figure 5.15: Traffic Capture - Burp Suite - 3

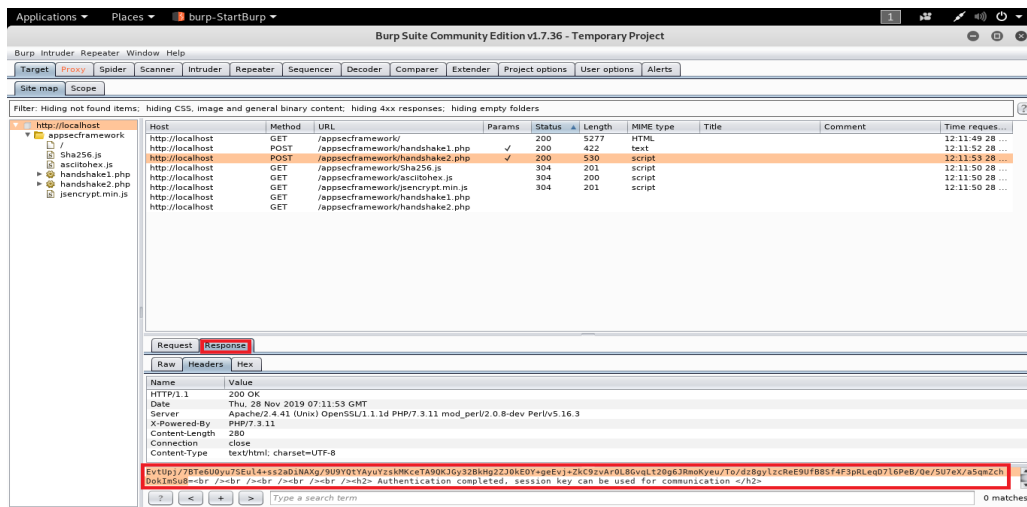


Figure 5.16: Traffic Capture - Burp Suite - 4

## 5.5.2 Functionality

Positive results were obtained using Scyther simulation and an implementation that was deployed in a client server model. Functionality presented was fully achieved. Strong authentication was achieved due to the exchanged certificates during the registration phase. This strong authentication was used as a basis for generating session keys. These session keys were then used for securing all future communications. The session key can be used to secure multiple requests without the need for renegotiating the shared secret. This results in persistent session, state fullness and authentication in a continuous manner. Renegotiation of the protocol allows forward secrecy to be achieved *i.e.* new secret keys are independently generated without depending on the previous keys. Access control mechanism were laid over the authenticated sessions to provision service in a restricted fashion *i.e.* secure provisioning.

### 5.5.3 Results

The presented scheme was successfully tested for the establishment of communication channels that are secure and support mutual authentication between both communicating parties *i.e.* the server and the client, without using TLS and TLS like technologies. The initial keys are exchanged over a secure side channel leveraging any out of band system that is commonly used for two factor authentications e.g. the server can send keys to the client using band email or SMS etc. Multiple sessions were successfully established. The scheme eliminated the need for any kind of cookies or tokens. On top of the authentication a basic model for access control restrictions can easily be implemented that can leverage the session information. This technique can be used in systems that do not have the typical security mechanisms available as well as the scheme can be overlaid on existing communication technologies.

## 5.6 Practical Adoption/Case Study

To illustrate the practicality of the proposed system, the following case study is presented.

Alice is running an e-commerce store. Bob is a regular customer and often purchases few things from Alice. The server is setup in a way that once the client creates an account on server providing two factors for verification like email and SMS, the server stores the client information like username, password, credit card number etc. Whenever the client wants to purchase something, client has to login and start purchasing, and the amount will be

deducted from the credit card. Bob wants to perform secure transactions in the current scenario. Following are the two possible assumptions based on the above case study that should be addressed to secure the communication channel.

**Assumption 1:** TLS is unavailable

TLS could be unavailable because of following two reasons:

- i) The server is not https secured.
- ii) A client is unable to establish secure session.

In both cases, communication is vulnerable to eavesdropping.

**Question:**

What happens if a secure server is not setup or client is unable to establish secure session because of device's misconfiguration or unavailability of PKI?

**Answer:**

Bob executed a transaction on a vulnerable connection. An attacker in the middle easily sniffed the packet. Now the attacker has access to Bob's authenticated information as well as the cookie which is valid for the next 7 days or 15 days depending on the implementation. Now the attacker can execute transactions masquerading as Bob until the cookie expires.

**Assumption 2:** TLS is available, but cookie and token information can be extracted. Attack is still possible even if TLS is available because of the following two reasons.

- i) Low entropy or not enough randomness used in token generation
- ii) Cookie leakage or server is not using "https" only cookies

In both cases, authenticated information is leaking.

**Question**

What happens if cookie and token information are leaking because of above reasons?

**Answer:**

TLS is available, but due to the lack of randomness in token generation and low entropy, the eavesdropper captures the packet and extract required information from it. Now the attacker has access information (session token/cookie) which can be used by an attacker to pretend as Bob for next transactions. In this scenario availability of TLS does not guarantee that the actual client is not authenticating.

**Solution to Problems:**

In future, Bob wants to establish a secure communication session with Alice in the unavailability of TLS and PKI. If Alice and Bob at the time of registration exchange implicit certificates (which only contains relevant information like email and public key). Both do not know each other, but after exchanging certificates, whenever Bob uses the already exchanged certificate, Alice can verify that it's the same user who has exchanged the certificate and vice versa. This way both Alice and Bob can verify each other and perform transactions until one party relocate the already exchanged certificate.

Using the scheme presented in this thesis, Alice and Bob can exchange certificates at the time of registration, verify each other once at the time of registration and then securely perform transactions in the unavailability of TLS and PKI.



## 5.7 Summary

In this chapter, the implementation of the solution and the results are discussed in detail which belongs to our objectives described at the beginning in chapter 1. The scheme is tested using a network protocol analyzing tool called “Scyther”. The traffic is also monitored using “Wireshark” and “Burp Suite” and found to be encrypted. In the next chapter conclusion of the thesis is given.

# Chapter 6

## Conclusion & Future Work

Chapter 6 concludes the presented thesis and highlights potential future research directions. It describes different research prospects of our research and identifies open research problems that still need to be solved by the research community.

### 6.1 Conclusion

Managing sessions and authenticating a valid user are both essential for the effective and efficient security of the system. Application development remains very sensitive to issues such as incorrect coding, logical errors, etc. Security systems that are based on TLS and SSL are secure, but these traditional protocols provide security to moving data only. They do not give security assurances to the systems which are subject to attacks that are not prevented by TLS and SSL. Since there is a diversity in information systems, this makes it hard to set up a unified security architecture.

In this research we have proposed a secure and effective mutual authentication scheme. This scheme extends the functionality of the registration phase to exchange already known authenticated information of the server and the client *i.e.* public keys. Authentication is achieved by the scheme using minimum transactions over the network. A shared secret is created at the end of a successful authentication. This shared secret can be used with different cipher suites. The proposed scheme depends on Message Authentication Code (MAC) and identification data that is affixed with every succeeding message to establish the authenticity of the message.

The attack vectors that are linked with tokens and cookies are eliminated because the proposed scheme does not use any of these as an authentication information transporter. The scheme is alternative for mutual authentication in networked applications which traditionally required public key infrastructure (PKI). The scheme is resistant to a range of attacks which were tested using Scyther. The attacks resisted are cookie theft, token forgery, session hijacking and man-in-the-middle.

## 6.2 Future Work

The future work that can be done with respect to this scheme is that hardware tokens can be studied in detail so that if they can be used as a source of authentication information for this scheme. As PGP involves initial key exchange, this scheme can also be looked at to be used with PGP and Blockchain. Android and IOS applications for secure communication can also be built using this scheme. For group communication, collaborative key

generation schemes can be studied in detail to use with the proposed scheme.

### **6.3 Summary**

This chapter has presented the conclusion of the thesis. Furthermore, it describes potential future directions in which this thesis can be extended for further research work.

# Bibliography

- [1] L. Catarinucci et al., “An IoT-Aware Architecture for Smart Healthcare Systems,” *IEEE Internet Things Journal*, volume 2, no. 6, pp. 515–526, Dec. 2015.
- [2] J. Lee, B. Bagheri, and H. A. Kao, “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems,” *Manufacturing Letters*, volume 3, pp. 18–23, Jan. 2015.
- [3] KM. Kenney, “CyberTerrorism in a Post-Stuxnet World,” *Orbis*, volume 59, no. 1, pp. 111–128, Jan. 2015.
- [4] E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3,” *RFC 8446* Aug. 2018.
- [5] E. Rescorla, “[Review] SSL and TLS: designing and building secure systems,” 2001. [Online]. Available: [http://dannypress.com/h/SSL\\_TLS.html](http://dannypress.com/h/SSL_TLS.html). [Accessed: 13-Nov-2019].
- [6] H. Y. Shih, H. L. Lu, C. C. Yeh, H. C. Hsiao, and S. K. Huang, “A generic web application testing and attack data generation method,” in *Advances in Intelligent Systems and Computing*, 2018, volume 733, pp. 232–247.

- [7] N. Nazir and M.K. Nazir, “American Scientific Research Journal for Engineering, Technology, and Sciences.,” *American Scientific Research Journal for Engineering, Technology, and Sciences*, volume 42, no. 1, pp. 166–187, Sep. 2018.
- [8] K. Zhao and L. Ge, “A Survey on the Internet of Things Security,” in 2013 *Ninth International Conference on Computational Intelligence and Security*, 2013, pp. 663–667.
- [9] A. One, “Smashing the stack for fun and profit,” *Phrack 49*, volume 7, no. 49, pp. 1996–11, 1996.
- [10] OWASP, “OWASP Top 10 - 2010: The Ten Most Critical Web Application Security Vulnerabilities,” 2010. [Online]. Available: <https://owasptop10.googlecode.com/files/OWASP%5CnTop%5Cn10%5Cn-%5Cn2010.pdf>. [Accessed: 01-Apr-2019].
- [11] P. De Ryck, L. Desmet, F. Piessens, and W. Joosen, “SecSess: keeping your session tucked in away in your browser,” in *Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC '15*, 2015, pp. 2171–2176.
- [12] Y. H. Lin et al., “SPATE: Small-group PKI-less authenticated trust establishment,” *IEEE Transaction on Mobile Computing*, volume 9, no. 12, pp. 1666–1681, Dec. 2010.
- [13] H. Birge-Lee, Y. Sun, A. Edmundson, J. Rexford, and P. Mittal, “Bamboozling Certificate Authorities with BGP,” in *USENIX Security Symposium (SEC)*, 2018, pp. 833–849.

- [14] D. Maimon, Y. Wu, M. McGuire, N. Stubler, and Z. Qiu, “SSL/TLS Certificates and Their Prevalence on the Dark Web — Venafi.” [Online]. Available: <https://www.venafi.com/sites/default/files/2019-02/Dark-Web-WP.pdf>. [Accessed: 19-Jun-2019].
- [15] J. Zhao, J. Bai, Q. Zhang et al., “The Discussion about Mechanism of Data Transmission in the OSI Model,” in *2018 International Conference on Transportation Logistics, Information Communication, Smart City (TLICSC 2018)*, 2018.
- [16] A. Al Abdulwahid, N. Clarke, I. Stengel, S. Furnell, and C. Reich, “Continuous and transparent multimodal authentication: reviewing the state of the art,” *Cluster Computing*, volume 19, no. 1, pp. 455–474, 2016.
- [17] K. J. Kim, C. G. Kim, T. K. Whangbo, and K. Yoon, “A continuous playing scheme on RESTful web service,” *Cluster Computing*, volume 19, no. 1, pp. 379–387, Mar. 2016.
- [18] K. Garrett, S. R. Talluri, and S. Roy, “On vulnerability analysis of several password authentication protocols,” *Innovations in Systems and Software Engineering*, volume 11, no. 3, pp. 167–176, Sep. 2015.
- [19] M. Abliz, “Internet Denial of Service Attacks and Defense Mechanisms,” University of Pittsburgh, March, 2011 p. 50.
- [20] F. Y. Yang, C. W. Hsu, and S. H. Chiu, “Password authentication scheme preserving identity privacy,” in *Proceedings - 2014 6th International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2014*, 2014, pp. 443–447.

- [21] S. H. Islam, “Design and analysis of an improved smartcard-based remote user password authentication scheme,” *International Journal of Communication Systems*, volume. 29, no. 11, pp. 1708–1719, Jul. 2016.
- [22] Q. Jiang, J. Ma, G. Li, and Z. Ma, “An improved password-based remote user authentication protocol without smart cards,” *Information Technology and Control*, volume. 42, no. 2, pp. 150–158, Jun. 2013.
- [23] Ding Wang, Chun-Guang Ma, Qi-Ming Zhang and Sendong Zhao, “Secure Password-based Remote User Authentication Scheme against Smart Card Security Breach”, *Journal of Networks*, Volume. 8, No. 1, PP. 148–155, Jan. 2013.
- [24] M. Lennartsson, “Evaluating the Memorability of Different Password Creation Strategies: A Systematic Literature Review,” 2019.
- [25] H.J. Mun, S. Hong, and J. Shin, “A novel secure and efficient hash function with extra padding against rainbow table attacks,” *Cluster Computing*, volume 21, no. 1, pp. 1161–1173, Mar. 2018.
- [26] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, “Zipf’s Law in Passwords,” *IEEE Transactions on Information Forensics and Security*, volume 12, no. 11, pp. 2776–2791, Nov. 2017.
- [27] J. Hirschberg and C. D. Manning, “Advances in natural language processing,” *Science*, volume 349, no. 6245. American Association for the Advancement of Science, pp. 261–266, 17-Jul-2015.



- [28] J. Bonneau, “The science of guessing: Analyzing an anonymized corpus of 70 million passwords,” in Proceedings - *IEEE Symposium on Security and Privacy*, 2012, pp. 538–552.
- [29] M. Dell’Amico, P. Michiardi, and Y. Roudier, “Password strength: An empirical analysis,” in Proceedings - *IEEE INFOCOM*, 2010, pp. 1–9.
- [30] “Unicode Consortium.” [Online]. Available: <http://www.unicode.org/>. [Accessed: 29-Jun-2019].
- [31] R. Amin and G. P. Biswas, “An Improved RSA Based User Authentication and Session Key Agreement Protocol Usable in TMIS,” *Journal of Medical Systems*, volume 39, no. 8, p. 79, 2015.
- [32] A. Witkovski, A. Santin, V. Abreu, and J. Marynowski, “An IdM and key-based authentication method for providing single sign-on in IoT,” in 2015 *IEEE Global Communications Conference, GLOBECOM 2015*, 2015, pp. 1–6.
- [33] Z. A. Alizai, N. F. Tareen, and I. Jadoon, “Improved IoT Device Authentication Scheme Using Device Capability and Digital Signatures,” in 2018 *International Conference on Applied and Engineering Mathematics (ICAEM)*, 2018, pp. 1–5.
- [34] L. Liu, J. Xu, C. Guo, K. Jiehui, S. Xu, and Z. Biao, “Exposing SQL Injection Vulnerability through Penetration Test based on Finite State Machine,” in 2016 *2nd IEEE International Conference on Computer and Communications, ICC 2016 - Proceedings*, 2017, pp. 1171–1175.

- [35] I. G. N. Mantra, M. Alaydrus, and H. M. Misni, “The web security and vulnerability analysis model on Indonesia Higher Education institution,” in *2016 International Conference on Informatics and Computing, ICIC 2016*, 2017, pp. 154–157.
- [36] C. Meshram, C.-C. Lee, C.-T. Li, and C.-L. Chen, “A secure key authentication scheme for cryptosystems based on GDLP and IFP,” *Soft Computing*, volume 21, no. 24, pp. 7285–7291, Dec. 2017.
- [37] D. Li, H. Chen, C. Zhong, T. Li, and F. Wang, “A New Self-Certified Signature Scheme Based on NTRUSing for Smart Mobile Communications,” *Wireless Personal Communications*, volume 96, no. 3, pp. 4263–4278, Oct. 2017.
- [38] S. Sciancalepore, G. Piro, G. Boggia, and G. Bianchi, “Public Key Authentication and Key Agreement in IoT Devices With Minimal Airtime Consumption,” *IEEE Embedded Systems Letters*, volume 9, no. 1, pp. 1–4, Mar. 2017.
- [39] A. Ometov, S. Bezzateev, N. Makitalo, S. Andreev, T. Mikkonen, and Y. Koucheryavy, “Multi-Factor Authentication: A Survey,” *Cryptography*, volume 2, no. 1, p. 1, Jan. 2018.
- [40] D. Wang and P. Wang, “Two Birds with One Stone: Two-Factor Authentication with Security beyond Conventional Bound,” *IEEE Transactions on Dependable and Secure Computing*, volume 15, no. 4, pp. 708–722, 2018.

- [41] Z. Li et al., “FBS-Radar: Uncovering Fake Base Stations at Scale in the Wild,” in *24th Network and Distributed System Security Symposium (NDSS)*, 2017.
- [42] M. Pannu, R. Bird, B. Gill, and K. Patel, “Investigating vulnerabilities in GSM security,” in *2015 International Conference and Workshop on Computing and Communication (IEMCON)*, 2015, pp. 1–7.
- [43] V. J. Rathod, N. C. Iyer, and S. M. Meena, “A survey on fingerprint biometric recognition system,” *Proceedings. 2015 International Conference on Green Computing and Internet of Things, ICGCIoT 2015*, pp. 323–326, 2016.
- [44] R. Gad, N. El-Fishawy, A. EL-SAYED, and M. Zorkany, “Multi-Biometric Systems: A State of the Art Survey and Research Directions,” *International Journal of Advanced Computer Science and Applications*, volume 6, no. 6, 2015.
- [45] Q. Jiang, J. Ma, G. Li, and X. Li, “Improvement of robust smart-card-based password authentication scheme,” *International Journal Communication Systems*, volume 28, no. 2, pp. 383–393, Jan. 2015.
- [46] A. X. Liu and L. A. Bailey, “PAP: A privacy and authentication protocol for passive RFID tags,” *Computer Communications*, volume 32, no. 7–10, pp. 1194–1199, May 2009.
- [47] W. M. AlOmari and H. Abusaimh, “Modified USB Security Token for User Authentication,” *Computer and Information Science*, volume 8, no. 3, 2015.

- [48] P. Gope, J. Lee, and T. Q. S. Quek, “Lightweight and Practical Anonymous Authentication Protocol for RFID Systems Using Physically Unclonable Functions,” *IEEE Transactions on Information Forensics and Security*, volume 13, no. 11, pp. 2831–2843, Nov. 2018.
- [49] D. Fox, “Hardware Security Module (HSM),” *Datenschutz und Datensicherheit - DuD*, volume 33, no. 9, pp. 564–564, Sep. 2009.
- [50] T. Caddy, S. W. Smith, and A. Stavrou, “Trusted Platform Module,” in *Encyclopedia of Cryptography and Security*, Boston, MA: Springer US, 2011, pp. 1332–1335.
- [51] TCG, “TPM Main Part 1 Design Principles,” *Specification version*, volume 1. p. 184, 2011.
- [52] Microsoft, “Trusted Platform Module Technology Overview (Windows 10) — Microsoft Docs,” Microsoft.com, 2017. [Online]. Available: <https://docs.microsoft.com/en-us/windows/security/information-protection/tpm/trusted-platform-module-overview>. [Accessed: 18-Nov-2019].
- [53] J. Jussila, “Http Cookie Weaknesses , Attack Methods and Defense Mechanisms: a Systematic Literature Review.” University of Jyvaskyla, 2018.
- [54] P. Kamal, “State of the Art Survey on Session Hijacking,” *Global Journal of Computer Science and Technology*, volume 16, no. 1, pp. 39–49, 2016.

- [55] S. Calzavara, A. Rabitti, and M. Bugliesi, “Dr Cookie and Mr Token - Web session implementations and how to live with them,” in *CEUR Workshop Proceedings*, 2018, volume 2058.
- [56] K. Lacroix, Y. L. Loo, and Y. B. Choi, “Cookies and Sessions: A Study of What They Are, How They Work and How They Can Be Stolen,” in *Proceedings - 2017 International Conference on Software Security and Assurance, ICSSA 2017*, 2018, pp. 20–24.
- [57] W. Bin Lee, H. B. Chen, S. S. Chang, and T. H. Chen, “Secure and efficient protection for HTTP cookies with self-verification,” *International Journal of Communication Systems*, volume 32, no. 2, 2019.
- [58] A. Amira, A. Ouadjaout, A. Derhab, and N. Badache, “Sound and Static Analysis of Session Fixation Vulnerabilities in PHP Web Applications,” in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy - CODASPY '17*, 2017, pp. 139–141.
- [59] A. Cahn, S. Alfeld, P. Barford, and S. Muthukrishnan, “An Empirical Study of Web Cookies,” in *Proceedings of the 25th International Conference on World Wide Web - WWW '16*, 2016, pp. 891–901.
- [60] X. Zheng, J. Jiang, H. Canada, and U. C. Berkeley, “Cookies Lack Integrity: Real-World Implications,” *Usenix Security*, pp. 707–721, 2015.
- [61] M. Dietz, A. Czeskis, and D. S. Wallach, “Origin-Bound Certificates: A Fresh Approach to Strong Client Authentication for the Web,” *Proceedings of the 21st USENIX conference on Security symposium*, pp. 317–331, 2012.

- [62] S.-M. Kim, Y.-H. Goo, M.-S. Kim, S.-G. Choi, and M.-J. Choi, “A method for service identification of SSL/TLS encrypted traffic with the relation of session ID and Server IP,” in 2015 *17th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2015*, pp. 487–490.
- [63] S. Rajasekar, P. Philominathan, and V. Chinnathambi, “Research Methodology”, 2013. [Online]. Available: <http://arxiv.org/pdf/physics/0601009.pdf>. [Accessed: 08-Oct-2019].
- [64] W. C. Booth, G. G. Colomb, and J. M. Williams, “The Craft of Research.” [Online]. Available: [http://sir.spbu.ru/en/programs/master/master\\_program\\_in\\_international\\_relations/digital\\_library/Book\\_Research\\_seminar\\_by\\_Booth.pdf](http://sir.spbu.ru/en/programs/master/master_program_in_international_relations/digital_library/Book_Research_seminar_by_Booth.pdf). [Accessed: 08-Oct-2019].
- [65] C. Woody, “Chapter 3: Research Methodology,” 2001. [Online]. Available: [https://shodhganga.inflibnet.ac.in/bitstream/10603/2026/16/16\\_chapter\\_3.pdf](https://shodhganga.inflibnet.ac.in/bitstream/10603/2026/16/16_chapter_3.pdf). [Accessed: 09-Oct-2019].
- [66] H. Gilbert and H. Handschuh, “Security analysis of SHA-256 and sisters,” International Workshop on Selected Areas in Cryptography, volume 3006, pp. 175–193, 2004.
- [67] S. Kelly and S. Frankel, “Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec,” *RFC 4868* May 2007.
- [68] C. Cremers, “Scyther tool,” 2016. [Online]. Available: <https://people.cispa.io/cas.cremers/scyther/>. [Accessed: 24-May-2019].

- [69] D. Dolev and A. C. Yao, “On the Security of Public Key Protocols,” *IEEE Transaction Information Theory*, volume 29, no. 2, pp. 198–208, Mar. 1983.
- [70] C. J. F. Cremers, S. Mauw, and E. P. de Vink, “Injective synchronisation: An extension of the authentication hierarchy,” *Theoretical Computer Science*, volume 367, no. 1–2, pp. 139–161, Nov. 2006.
- [71] D. McGrew and D. Bailey, “AES-CCM Cipher Suites for Transport Layer Security (TLS),” *RFC 6655* Jul-2012.
- [72] J. Salowey, A. Choudhury, and D. McGrew, “AES Galois Counter Mode (GCM) Cipher Suites for (TLS),” *RFC 5288*, pp. 1–8, Aug. 2008.

# Appendix A

## Scyther

**File name: appsec.spdl**

/\*This program and associated code were created to simulate the research titled “Key-based Cookie-less Session Management Framework for Application Layer Security” published in IEEEAccess, DOI: 10.1109/ACCESS.2019.2940331. The author of the research retains all the rights to this program and associated source code. This program and associated source code are distributed under the GNU General Public License. Any third-party modules that were used in the creation of this program and its source code are the property of their respective owners and are governed by their respective licenses.

Copyright (C) <2019><Authors: Zahoor Ahmed Alizai & Malik Hamza Murtaza>

This program is free software: you can redistribute it and/or modify it un-



der the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see [<https://www.gnu.org/licenses/>.\\*](https://www.gnu.org/licenses/)

```

usertype Timestamp;
hashfunction H;
protocol handshake(Server, Client)
{
  role Server
  {
    fresh ServerRandom: Nonce;
    fresh ServerTimestamp: Timestamp;
    send_1(Server,Client,{Server,ServerRandom,ServerTimestamp}pk(Client));
    var ClientRandom: Nonce;
    recv_2(Client,Server,H(Client,ServerRandom,ServerTimestamp,
    ClientRandom),{ClientRandom}pk(Server));
  }
  //Server Sending data to client after handshake is complete.

```

```

fresh ServerData: Nonce;
send_3(Server,Client,{Server,ServerData}H(ServerRandom,ClientRandom));
//Server Receiving Client's Data.
var ClientData: Nonce;
recv_4(Client,Server,{ClientData}H(ServerRandom,ClientRandom));
claim_Server1(Server,Secret,Timestamp);
claim_Server2(Server,Secret,ServerRandom);
claim_Server3(Server,Secret,ClientRandom);
claim_Server4(Server,Secret,ServerData);
claim_Server5(Server,Secret,ClientData);
claim_Server6(Server,Niagree);
claim_Server7(Server,Nisynch);
claim_Server8(Server,Alive);
claim_Server9(Server,Weakagree);
};
role Client
{
var ServerRandom: Nonce;
var ServerTimestamp: Timestamp;
recv_1(Server,Client,{Server,ServerRandom,ServerTimestamp}pk(Client));
fresh ClientRandom: Nonce;
send_2(Client,Server,H(Client,ServerRandom,ServerTimestamp,
ClientRandom),{ClientRandom}pk(Server));
//Receive server data
var ServerData: Nonce;

```

```
recv_3(Server,Client,{Server,ServerData}H(ServerRandom,ClientRandom));
//Client sends data to server
fresh ClientData: Nonce;
send_4(Client,Server,{ClientData}H(ServerRandom,ClientRandom));
claim_Client1(Client,Secret,Timestamp);
claim_Client2(Client,Secret,ClientRandom);
claim_Client3(Client,Secret,ServerRandom);
claim_Client4(Client,Secret,ClientData);
claim_Client5(Client,Secret,ServerData);
claim_Client6(Client,Niagree);
claim_Client7(Client,Nisynch);
claim_Client8(Client,Alive);
claim_Client9(Client,Weakagree);
};
};
```

# Appendix B

## Java

**File name: zatpclient.java**

/\*This program and associated code were created to simulate the research titled “Key-based Cookie-less Session Management Framework for Application Layer Security” published in IEEEAccess, DOI: 10.1109/ACCESS.2019.2940331. The author of the research retains all the rights to this program and associated source code. This program and associated source code are distributed under the GNU General Public License. Any third-party modules that were used in the creation of this program and its source code are the property of their respective owners and are governed by their respective licenses.

Copyright (C) <2019><Authors: Zahoor Ahmed Alizai & Malik Hamza Murtaza>

This program is free software: you can redistribute it and/or modify it un-

der the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see [<https://www.gnu.org/licenses/>.\\*](https://www.gnu.org/licenses/)

```
package zatpclient;
import java.net.*;
import java.io.*;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;
import java.util.Random;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.JTextArea;
```

```
import zatplib.*;

public class ZatpClient
{
    private static final String SELFID = "zahoor";
    private static Socket connectionSocket;
    private static Scanner recDataFromServer;
    private static PrintStream sendDataToServer;
    private static boolean connected = false;
    private static boolean sessionEstablished = false;
    private static SecretKey AESKey = null;
    public static void connectToServer(String serverAddress)
    {
        try
        {
            //init connection to server
            connectionSocket = new Socket(serverAddress,4567);
            //init write to / read from server streams
            recDataFromServer = new Scanner(new InputStreamReader
            (connectionSocket.getInputStream()));
            sendDataToServer = newPrintStream(connectionSocket.getOutputStream());
            connected = true;
        }
        catch (IOException ex)
        {
            connected = false;
        }
    }
}
```

```
Logger.getLogger(ZatpClient.class.getName()).log(Level.SEVERE, null, ex);
}
}
public static void disconnectFromServer(JTextArea loggerArea)
{
try
{
AESKey = null;
connectionSocket.close();
recDataFromServer.close();
sendDataToServer.close();
sessionEstablished = false;
connected = false;
loggerArea.setText("");
}
catch (IOException ex)
{
Logger.getLogger(ZatpClient.class.getName()).log(Level.SEVERE, null, ex);
}
}
public static boolean getConnectionStatus()
{
return connected;
}
public static boolean getSessionStatus()
```

```
{
return sessionEstablished;
}

public static void negotiateProtocol(JTextArea loggerArea)
{
//load personal key pair
user self = userManager.retrieveUserInformation(SELFID);
//start protocol
String dataFromServer;
//send user id to server
sendDataToServer.println(self.getUsername());
//wait for server's response
dataFromServer = recDataFromServer.nextLine();
//check if the server acknowledges to user
if(dataFromServer.contains("KCA"))
{
loggerArea.append("Server Could Not Find The User.");
}
else
{
try
{
//decrypt recieved challenge
loggerArea.append("(Challenge Recieved) " + dataFromServer + "");
byte[] challenge = self.decrypt(Base64.getDecoder().decode(dataFromServer));
```



```
//decrypted challange
String decryptedChallengeString = new String(challenge, "UTF8");
loggerArea.append("(Decrypted Challenge) " + decryptedChallenge
String+ "");
//generate client side random
Random rand = new Random();
long clientsRandom = rand.nextLong();
loggerArea.append("(User's Random) " + clientsRandom+ "");
//concat client's random with recieved string
String responseToChallenge = decryptedChallengeString+ ""+clients
Random;
loggerArea.append("(Challenge Responce) " +responseToChallenge+ "");
//create responce hash from concatenated string
String responseHash = Base64.getEncoder().encodeToString
(MessageDigest.getInstance("SHA-256").digest(responseToChallenge.
getBytes()));
/*
* load server's public key
*/
user server = userManager.retrieveUserInformation("server");
String encryptedClientRandom = Base64.getEncoder().encodeToString
(server.encrypt((""+clientsRandom).getBytes()));
//send the base64 hash and the base64 encrypted client's random
(server's PU) back to server for authentication.
//client random will be decrypted on server side, and the concatenated
```

string will be recreated.

//the concatenated string will be hashed and used as a proof of authentication.

```
sendDataToServer.println(encryptedClientRandom);
sendDataToServer.println(responseHash);
dataFromServer = recDataFromServer.nextLine();
if(dataFromServer.contains("KCA"))
{
loggerArea.append("AUTHENTICATION FAILED!");
}
if(dataFromServer.contains("ACK"))
{
String str = decryptedChallangeString.substring(0,decryptedChallange
String.indexOf(""))+"."+clientsRandom;
loggerArea.append("(KeyGenString)" + str + "");
byte[] bAESKey = MessageDigest.getInstance("SHA-256").digest
(str.getBytes());
String sAESKey = Base64.getEncoder().encodeToString(bAESKey);
loggerArea.append("Protocol Negotiated, Setting up session key.");
loggerArea.append("(Session Key) " + sAESKey + "");
AESKey = new SecretKeySpec(bAESKey, 0, bAESKey.length, "AES");
loggerArea.append("Secure Channel Established.
Proceed With Requests To The Server.");
sessionEstablished = true;
}
```

```
}  
catch (IOException — NoSuchAlgorithmException ex)  
{  
    Logger.getLogger(ZatpClient.class.getName()).log(Level.SEVERE, null, ex);  
}  
}  
}  
}
```

**File name: zatpserver.java**

/\*This program and associated code were created to simulate the research titled “Key-based Cookie-less Session Management Framework for Application Layer Security” published in IEEEAccess, DOI: 10.1109/ACCESS.2019.2940331. The author of the research retains all the rights to this program and associated source code. This program and associated source code are distributed under the GNU General Public License. Any third-party modules that were used in the creation of this program and its source code are the property of their respective owners and are governed by their respective licenses.

Copyright (C) <2019><Authors: Zahoor Ahmed Alizai & Malik Hamza Murtaza>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free

Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see [<https://www.gnu.org/licenses/>.\\*](https://www.gnu.org/licenses/)

```
package zatpserver;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.security.MessageDigest;
import java.sql.Timestamp;
import java.util.Base64;
import java.util.Random;
import java.util.Scanner;
import zatplib.*;
public class ZatpServer
{
private static final String SELFID = "server";
```

```
public static void main(String args[]) throws Exception
{
    //create server and listen for connections
    ServerSocket serverSocket = new ServerSocket(4567);
    Socket connectionSocket = serverSocket.accept();
    //init write to / read from client streams
    Scanner recDataFromClient = new Scanner(new
    InputStreamReader(connectionSocket.getInputStream()));
    PrintStream sendDataToClient = new PrintStream(connectionSocket
    .getOutputStream());
    //load personal key pair
    user self = userManager.retrieveUserInformation(SELFID);
    //start protocol
    String dataFromClient;
    //receive user's id
    dataFromClient = recDataFromClient.nextLine();
    System.out.println("User ' "+dataFromClient+" ' wants to authenticate.");
    //look for the user
    user client = userManager.retrieveUserInformation(dataFromClient);
    //if client info not found, drop request
    if(client==null)
    {
        sendDataToClient.println("KCA");
        serverSocket.close();
        connectionSocket.close();
    }
}
```

```
recDataFromClient.close();
sendDataToClient.close();
return;
}
//else continue negotiation
System.out.println("username = " + client.getUsername());
System.out.println("public key = " + client.getBase64PublicKey());
//sending encrypted challange + timestamp
Random rand = new Random();
long serversRandom = rand.nextLong();
Timestamp ts = new Timestamp(System.currentTimeMillis());
String challange = serversRandom+" "+ts;
System.out.println("sending challange = " + challange);
byte[] cipherData = client.encrypt((challange).getBytes());
String challangeString = Base64.getEncoder().encodeToString(cipherData);
sendDataToClient.println(challangeString);
System.out.println("Challenge Sent = " + challangeString);
//recieve clients random
String sEncryptedClientsRandom = recDataFromClient.nextLine();
String sClientsRandom = new String(self.decrypt(Base64.getDecoder().
decode(sEncryptedClientsRandom)), "UTF8");
System.out.println("Clients Random is = "+sClientsRandom);
//recieve clients authentichash
String sRecievedAuthenticHash = recDataFromClient.nextLine();
//recreate authentic hash for authentication
```

```
String recreatedAuthenticHash = Base64.getEncoder().encodeToString
(MessageDigest.getInstance("SHA-256").digest((challenge+ "" +sClients
Random).getBytes()));
System.out.println("recieved auth hash : " +sRecievedAuthenticHash);
System.out.println("recreated auth hash : " +recreatedAuthenticHash);
if(sRecievedAuthenticHash.compareTo(recreatedAuthenticHash) == 0)
{
sendDataToClient.println("ACK");
System.out.println("Successfully Authenticated!");
}
else
{
sendDataToClient.println("KCA");
System.out.println("AUTHENTICATION FAILED!");
serverSocket.close();
connectionSocket.close();
recDataFromClient.close();
sendDataToClient.close();
return;
}
serverSocket.close();
connectionSocket.close();
recDataFromClient.close();
sendDataToClient.close();
}
```

```
}
```

Two files in zatplib (library folder).

File name: user.java

/\*This program and associated code were created to simulate the research titled “Key-based Cookie-less Session Management Framework for Application Layer Security” published in IEEEAccess, DOI: 10.1109/ACCESS.2019.2940331. The author of the research retains all the rights to this program and associated source code. This program and associated source code are distributed under the GNU General Public License. Any third-party modules that were used in the creation of this program and its source code are the property of their respective owners and are governed by their respective licenses.

Copyright (C) <2019><Authors: Zahoor Ahmed Alizai & Malik Hamza Murtaza>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY-



ITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see [<https://www.gnu.org/licenses/>.\\*](https://www.gnu.org/licenses/)

```
package zatplib;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.util.Base64;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
public class user
{
private String userName = null;
private PrivateKey userPrivateKey = null;
private PublicKey userPublicKey = null;
Cipher encryptCipher = null;
Cipher decryptCipher = null;
```

```
user(String Id, PublicKey PU)
{
    userName = Id;
    userPublicKey = PU;
    try
    {
        this.encryptCipher = Cipher.getInstance("RSA");
        this.encryptCipher.init(Cipher.ENCRYPT_MODE, userPublicKey);
    }
    catch (NoSuchAlgorithmException — NoSuchPaddingException — Invalid-
    KeyException ex)
    {
        Logger.getLogger(user.class.getName()).log(Level.SEVERE, null, ex);
    }
}

user(String Id, PrivateKey PK)
{
    userName = Id;
    userPrivateKey = PK;
    try
    {
        this.decryptCipher = Cipher.getInstance("RSA");
        this.decryptCipher.init(Cipher.DECRYPT_MODE, userPrivateKey);
    }
    catch (NoSuchAlgorithmException — NoSuchPaddingException — Invalid-
```

```
KeyException ex)
{
    Logger.getLogger(user.class.getName()).log(Level.SEVERE, null, ex);
}
}

user(String Id, PrivateKey PK, PublicKey PU)
{
    userName = Id;
    userPublicKey = PU;
    userPrivateKey = PK;
    try
    {
        this.encryptCipher = Cipher.getInstance("RSA");
        this.decryptCipher = Cipher.getInstance("RSA");
        this.encryptCipher.init(Cipher.ENCRYPT_MODE, userPublicKey);
        this.decryptCipher.init(Cipher.DECRYPT_MODE, userPrivateKey);
    }
    catch (NoSuchAlgorithmException — NoSuchPaddingException — Invalid-
        KeyException ex)
    {
        Logger.getLogger(user.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public String getUsername()
{
```

```
return userName;
}
public boolean publicKeyExists()
{
return userPublicKey != null;
}
public PublicKey getPublicKey()
{
return userPublicKey;
}
public String getBase64PublicKey()
{
return Base64.getEncoder().encodeToString(userPublicKey.getEncoded());
}
public boolean privateKeyExists()
{
return userPrivateKey != null;
}
public PrivateKey getPrivateKey()
{
return userPrivateKey;
}
public String getBase64PrivateKey()
{
return Base64.getEncoder().encodeToString(userPrivateKey.getEncoded());
```

```
}  
public byte[] encrypt(byte[] data)  
{  
    byte[] returnData = null;  
    try  
    {  
        returnData = encryptCipher.doFinal(data);  
    }  
    catch (IllegalBlockSizeException — BadPaddingException ex)  
    {  
        Logger.getLogger(user.class.getName()).log(Level.SEVERE, null, ex);  
    }  
    return returnData;  
}  
public byte[] decrypt(byte[] data)  
{  
    byte[] returnData = null;  
    try  
    {  
        returnData = decryptCipher.doFinal(data);  
    }  
    catch (IllegalBlockSizeException — BadPaddingException ex)  
    {  
        Logger.getLogger(user.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

```
return returnData;  
}  
}
```

**File name: userManager.java**

/\*This program and associated code were created to simulate the research titled “Key-based Cookie-less Session Management Framework for Application Layer Security” published in IEEEAccess, DOI: 10.1109/ACCESS.2019.2940331. The author of the research retains all the rights to this program and associated source code. This program and associated source code are distributed under the GNU General Public License. Any third-party modules that were used in the creation of this program and its source code are the property of their respective owners and are governed by their respective licenses.

Copyright (C) <2019><Authors: Zahoor Ahmed Alizai & Malik Hamza Murtaza>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General

Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see [<https://www.gnu.org/licenses/>.\\*](https://www.gnu.org/licenses/)

```
package zatplib;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.PrintWriter;
import java.security.KeyFactory;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
public class userManager
{
```

```
/* public static void main(String[] args)
{
createUserInformation("server");
createUserInformation("zahoor");
*/
public static user retrieveUserInformation(String Id)
{
File userInfoFile = new File(Id+".key");
if(userInfoFile.exists())
{
try
{
Scanner userInfoFileReader = new Scanner(new File(Id+".key"));
String keyFileStatusFlag = userInfoFileReader.nextLine();
System.out.println("key file status = " + keyFileStatusFlag);
int flagStatus = Integer.parseInt(keyFileStatusFlag);
String sPK, sPU;
byte[] bPK, bPU;
PublicKey PU;
PrivateKey PK;
switch(flagStatus)
{
case 1:
sPU = userInfoFileReader.nextLine();
bPU = Base64.getDecoder().decode(sPU);
```



```
PU = KeyFactory.getInstance("RSA").generatePublic(new X509
EncodedKeySpec(bPU));
return new user(Id, PU);
case 2:
sPK = userInfoFileReader.nextLine();
bPK = Base64.getDecoder().decode(sPK);
PK = KeyFactory.getInstance("RSA").generatePrivate(new PKCS8
EncodedKeySpec(bPK));
return new user(Id, PK);
case 3:
sPU = userInfoFileReader.nextLine();
sPK = userInfoFileReader.nextLine();
bPU = Base64.getDecoder().decode(sPU);
PU = KeyFactory.getInstance("RSA").generatePublic(new X509Encoded
KeySpec(bPU));
bPK = Base64.getDecoder().decode(sPK);
PK = KeyFactory.getInstance("RSA").generatePrivate(new PKCS8
EncodedKeySpec(bPK));
return new user(Id, PK, PU);
default:
break;
}
}
catch (NoSuchAlgorithmException — InvalidKeySpecException — FileNot-
FoundException ex)
```

```
{
Logger.getLogger(userManager.class.getName()).log(Level.SEVERE, null, ex);
}
}
return null;
}
```

```
public static void createUserInformation(String Id)
{
try
{
//create a file for the user specified as the argument
PrintWriter userInfoFile = new PrintWriter(Id+ ".key");
//create a new rsa key pair for the user
KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");
kpg.initialize(1024);
KeyPair kp = kpg.generateKeyPair();
PublicKey PU = kp.getPublic();
PrivateKey PK = kp.getPrivate();
userInfoFile.println(3);
userInfoFile.println(Base64.getEncoder().encodeToString(PU.getEncoded()));
userInfoFile.println(Base64.getEncoder().encodeToString(PK.getEncoded()));
userInfoFile.flush();
userInfoFile.close();
}
}
```

```
catch (IOException — NoSuchAlgorithmException ex)
{
    Logger.getLogger(userManager.class.getName()).log(Level.SEVERE, null, ex);
}
}

public static void createUserInformation(user userObject)
{
    try
    {
        //create a file for the user specified as the argument
        PrintWriter userInfoFile = new PrintWriter(userObject.getUsername()+“.key”);
        if(userObject.privateKeyExists() && userObject.publicKeyExists())
        {
            PublicKey PU = userObject.getPublicKey();
            PrivateKey PK = userObject.getPrivateKey();
            userInfoFile.println(3);
            userInfoFile.println(Base64.getEncoder().encodeToString(PU.getEncoded()));
            userInfoFile.println(Base64.getEncoder().encodeToString(PK.getEncoded()));
        }
        if(!userObject.privateKeyExists() && userObject.publicKeyExists())
            PublicKey PU = userObject.getPublicKey(); userInfoFile.println(1);
            userInfoFile.println(Base64.getEncoder().encodeToString(PU.getEncoded()));
        }
        if(userObject.privateKeyExists() && !userObject.publicKeyExists())
        {
```

```
PrivateKey PK = userObject.getPrivateKey();
userInfoFile.println(2);
userInfoFile.println(Base64.getEncoder().encodeToString(PK.getEncoded()));
}
}
catch (IOException ex)
{
Logger.getLogger(userManager.class.getName()).log(Level.SEVERE, null, ex);
}
}
}
```

# Appendix C

## PHP (Server-side)

**File name: index.html**

<!--This program and associated code were created to simulate the research titled “Key-based Cookie-less Session Management Framework for Application Layer Security” published in IEEEAccess, DOI: 10.1109/ACCESS.2019.2940331.

The author of the research retains all the rights to this program and associated source code. This program and associated source code are distributed under the GNU General Public License. Any third-party modules that were used in the creation of this program and its source code are the property of their respective owners and are governed by their respective licenses.

Copyright (C) <2019><Authors: Zahoor Ahmed Alizai & Malik Hamza Murtaza>

This program is free software: you can redistribute it and/or modify it un-

der the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see [<https://www.gnu.org/licenses/>.\\*/->](https://www.gnu.org/licenses/)

```
<!DOCTYPE html>
<html>
<head>
<script src="jsencrypt.min.js" ></script>
<script src="Sha256.js" ></script>
<script src="asciitohex.js" ></script>
<script type="text/javascript">
function buf2hex(buffer)
{
// buffer is an ArrayBuffer
// create a byte array (Uint8Array) that we can use to read the array buffer
const byteArray = new Uint8Array(buffer);
// for each element, we want to get its two-digit hexadecimal representation
```

```
const hexParts = [];  
for(let i = 0; i <byteArray.length; i++)  
{  
  // convert value to hexadecimal  
  const hex = byteArray[i].toString(16);  
  // pad with zeros to length 2  
  const paddedHex = ('00' + hex).slice(-2);  
  // push to array  
  hexParts.push(paddedHex);  
}  
// join all the hex values of the elements into a single string  
return hexParts.join("");  
}  
// Check for the various File API support.  
function loadDoc()  
{  
  if (window.File window.FileReader window.FileList window.Blob)  
  {  
    document.getElementById("fileapi_status").innerHTML = "File API Sup-  
ported";  
  }  
  else  
  {  
    document.getElementById("fileapi_status").innerHTML = "File API Not Sup-  
ported";  
  }  
}
```

```
var xhttp = new XMLHttpRequest();
var clientid = "email@email.email";
xhttp.open("POST", "handshake1.php", false);
xhttp.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
xhttp.send("email="+clientid);
document.getElementById("current_status").innerHTML =
xhttp.responseText;
//start decrypting challenge
var decrypt = new JSEncrypt();
decrypt.setPrivateKey(document.getElementById("privkey").value);
var uncrpyted = decrypt.decrypt(xhttp.responseText);
document.getElementById("current_status").innerHTML = document
.getElementById("current_status").innerHTML + "<br /><h2>Unencrypted
Challenge:</h2>" + uncrpyted;
//create client's 128 bit random number and hex encode it
var array = new Uint8Array(16);
window.crypto.getRandomValues(array);
var clientrandom = buf2hex(array.buffer);
document.getElementById("current_status").innerHTML = document
.getElementById("current_status").innerHTML + "<br />
<h2>clientrandom:</h2>" + clientrandom;
//produce the client's proof of successfull decryption
var clientproof = uncrpyted+":"+clientrandom+":"+clientid;
```



```
document.getElementById("current_status").innerHTML = document
.getElementById("current_status").innerHTML + "<br /><h2>
clientproof plain text:</h2>" + clientproof;
var digestHex = SHA256(clientproof);
document.getElementById("current_status").innerHTML = document
.getElementById("current_status").innerHTML + "<br /><h2>hash:</h2>"
+ digestHex;
//encrypt client random using server's public key
var encrypt = new JSEncrypt();
encrypt.setPublicKey(document.getElementById("serverpukey").value);
var encryptedclientrandom = encrypt.encrypt(clientrandom);
document.getElementById("current_status").innerHTML = document
.getElementById("current_status").innerHTML + "<h2>client's encrypted
random </h2>" + encryptedclientrandom;
document.getElementById("current_status").innerHTML = document
.getElementById("current_status").innerHTML + "<br /><h2>sending
challengehash:{clientrandom}PUserver to server</h2>";
//send digest and clientrandom to the server
var challangeresponce = "email=" + clientid + "digest=" + digestHex + "
&clientrandom=" +
toHex(encryptedclientrandom);
var handshake2request = new XMLHttpRequest();
handshake2request.open("POST", "handshake2.php", false);
handshake2request.setRequestHeader("Content-type", "application/x-www-
form-urlencoded");
```

```

handshake2request.send(challangeresponce);
document.getElementById("current_status").innerHTML = document
.getElementById("current_status").innerHTML + "<br />" +
handshake2request.responseText;
}
</script>
</head>
<body onload="loadDoc()" >
<div id="status_fileapi" >
<h2 id="fileapi_status" ></h2>
</div>
<div id="status_div" >
<h2Value:</h2>
<div id="current_status" ></div>
</div>
<br />
<div id="encryptedcontent" >
</div>
<textarea id="privkey" style="display:none;" rows="15" cols="65" >
—BEGIN RSA PRIVATE KEY—
MIICXAIBAAKBgQCZlke4Y2R+Tj9pdKI9jxpPfHuVzuke/inoPIYk6D
/AGoNmSNRGN8/zlSsJIIXCP0Z6L4AI3WiBY1ZEzayhYSniN1eGD9/h
96Z6jsVrMgUSz3fCAUkL0qKyF0OLu5D6ZJt1s8WErvHiuacYbk7RKA
l7mZGJYq0wwnBunavMALXwIDAQABoGAanfH+fF0vZYSoVEIEvJ7
w7RAm8YWlrSMaoBiYX1ajBoVErft52VTU8EJV5fM7a4ddiAtene1Sm5

```

```
c4O3P0gt6u+qnc4DzaoyPwywqWlJA337Ppt8vUBbAGqP8+IeY6emG++
K5GzYrZrhG/GKqVyp+gcOGsyoRHDEeki/I7R/74MkCQQD5K2h7p31D2d
z/NqmDbYdk/Z4IBm6AGIDdq4oHmQF/C2ICLYV0VkQbY6W0IWGKAH5
+zxMWcNpWl1xnUArMLCN7AkEAncwaxB4YNUxgTYvn322ZC2Duvgr2z
/kC2w8Hyaws2UswlUtvGDE3WD3uOd8A44rungKho0COykgK+IP5PRPQ
bQJAWCrp/8dWZenzb0NSXDUnka7EeqZ790u0XbvTTbjwdJn8hjTBYlccJ
mzuN6YcK47dM9XmNSydtcI9ajlCeNitfQJACUUWRXMnJOppteSEKKH9
naeCHvPx7+HHAyts37IXqGQ8ZGjcEgHSKILq3cd1++gLgIjTCg4e1U9KJc6
NE8N4OQJBAK479nWDCEMiI/fxeQwIfzmiGwTDN2hDL3A4aXQQzzNRM0
TJ7CVNAYjfK/+9k0lMLuGxJtiSU8wQuoo/kP2FcJI=
```

```
—END RSA PRIVATE KEY—
```

```
</textarea><br/>
```

```
<textarea id="serverpukey" style="display:none;" rows="15" cols="65">
```

```
—BEGIN PUBLIC KEY—
```

```
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCNIWh1PdM9o
VkVH6VVOgSQBPiZNVQ+YS994n0fLdPGcvt90j0VQQSQvEBB+Dwn
AzY+wxWEm0HxefHbN5jhr3Tp/PGb+EGLT6zfxHvuzXCdvtN/0f6
AzBoTSeo0pf1QD4u8cu7aqJeKBX9/f99d8TGWcfsH1TgnVUmqTgSIGXI7
wIDAQAB
```

```
—END PUBLIC KEY—
```

```
</textarea><br/></body>
```

```
</html>
```

**File name: handshake1.php**

```
<?php
/*This program and associated code were created to simulate the research ti-
tled “Key-based Cookie-less Session Management Framework for Application
Layer Security” published in IEEEAccess, DOI: 10.1109/ACCESS.2019.2940331.
The author of the research retains all the rights to this program and associ-
ated source code. This program and associated source code are distributed
under the GNU General Public License. Any third-party modules that were
used in the creation of this program and its source code are the property of
their respective owners and are governed by their respective licenses.
```

```
Copyright (C) <2019><Authors: Zahoor Ahmed Alizai & Malik Hamza
Murtaza>
```

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see [<https://www.gnu.org/licenses/>](https://www.gnu.org/licenses/).\*/

```
//receive User ID
$email = $_POST["email"];
//Check for corresponding public key in database
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "registration";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error)
{
die("Connection failed: " . $conn->connect_error);
}
//find id in db
$sql = "SELECT email, pukey FROM users WHERE email = '$email'";
$result = $conn->query($sql);
//retrieve corresponding Public key
if ($result->num_rows == 1) {
// output data of each row
while($row = $result->fetch_assoc())
{
$pukey = $row["pukey"];
//send challenge
```

```
$bytes = openssl_random_pseudo_bytes(16, $cstrong);
$random_by_Server = bin2hex($bytes);
$dateTime = new DateTime();
$timestamp = $dateTime->Timestamp();
$unencryptedchallenge = $servername.":".$random_by_Server.":".$timestamp;
$userrecordfile = fopen($email.".txt", "w") or die("Unable to open file!");
fwrite($userrecordfile, $unencryptedchallenge);
fclose($userrecordfile);
if (!$publicKey = openssl_pkey_get_public($pukey)) die('Loading Public Key
failed');
$encryptedchallenge = "";
if (!openssl_public_encrypt($unencryptedchallenge, $encryptedchallenge, $pub-
licKey)) die('Failed to encrypt data');
$encryptedchallenge = base64_encode($encryptedchallenge);
echo $encryptedchallenge;
}
}
else if
($result->num_rows <1)
{
echo "0 results";
}
else
{
echo "too many results";
```

```
}  
$conn->close();  
?>
```

**File name: handshake2.php**

/\*This program and associated code were created to simulate the research titled “Key-based Cookie-less Session Management Framework for Application Layer Security” published in IEEEAccess, DOI: 10.1109/ACCESS.2019.2940331. The author of the research retains all the rights to this program and associated source code. This program and associated source code are distributed under the GNU General Public License. Any third-party modules that were used in the creation of this program and its source code are the property of their respective owners and are governed by their respective licenses.

Copyright (C) <2019><Authors: Zahoor Ahmed Alizai & Malik Hamza Murtaza>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General

Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see [<https://www.gnu.org/licenses/>.\\*](https://www.gnu.org/licenses/)

```
<?php
$digest = $_POST["digest"];
$encryptedclientrandom = $_POST["clientrandom"];
$email = $_POST["email"];
echo "2>Server is verifying response for ". $email. "</h2>";
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "registration";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error)
{
die("Connection failed: " . $conn->connect_error);
}
//find id in db
$sql = "SELECT * FROM serverinfo WHERE clientid = '$email'";
$result = $conn->query($sql);
//retrieve corresponding Public key
```



```
$clientrandom = "";
if ($result->num_rows == 1)
{
// output data of each row
while($row = $result->fetch_assoc())
{
echo hex2bin($encryptedclientrandom)."/>";
$serverprikeytext=$row["serverprikey"];
$serverprikey = openssl_pkey_get_private($serverprikeytext);
if (!openssl_private_decrypt(base64_decode(hex2bin($encryptedclientrandom)),
$clientrandom, $serverprikey)) die('Failed to encrypt data');
try
{
$myfile = fopen($email.".txt", "r") or die("Handshake phase 1 incomplete!");
$clientproofcheck = fgets($myfile).":".$clientrandom.":".$email;
echo "<br />".$clientproofcheck;
$recheckhash = hash('sha256',$clientproofcheck);
echo "<br />". $recheckhash;
if($digest == $recheckhash)
{
echo "<h1 >Authentication Completed, proceed with session key establish-
ment </h1>";
}
else
{
```

```
die("Authentication Failed");  
}  
catch (Exception $e)  
{  
echo "Handshake phase 1 incomplete!";  
}  
}  
}  
?>
```

# Appendix D

## JavaScript (Client-side)

1) A Sha256 library used in this research is developed by Angel Marin, Paul Johnston. Library code is available on:

[http://www.webtoolkit.info/javascript\\_sha256.html#.XdeuougzIU](http://www.webtoolkit.info/javascript_sha256.html#.XdeuougzIU)

2) Libraries used for encryption/decryption and key generation in this research is available on:

<https://github.com/travist/jsencrypt>

3) A library used for ascii to hex conversion is available on:

<https://gist.github.com/valentinkostadinov/5875467>