Mobile User Authentication System (MUAS) for E-commerce Applications

By

Rania A. Molla

A thesis submitted

In partial fulfilment of the requirements of

Edinburgh Napier University,

For the award of Doctor of Philosophy

April 2017

Abstract

The rapid growth of e-commerce has many associated security concerns. Thus, several studies to develop secure online authentication systems have emerged. Most studies begin with the premise that the intermediate network is the primary point of compromise. In this thesis, we assume that the point of compromise lies within the end-host or browser; this security threat is called the man-in-the-browser (MITB) attack. MITB attacks can bypass security measures of public key infrastructures (PKI), as well as encryption mechanisms for secure socket layers and transport layer security (SSL/TLS) protocol. This thesis focuses on developing a system that can circumvent MITB attacks using a two-phase secure-user authentication system, with phases that include challenge and response generation. The proposed system represents the first step in conducting an online business transaction.

The proposed authentication system design contributes to protect the confidentiality of the initiating client by requesting minimal and non-confidential information to bypass the MITB attack and transition the authentication mechanism from the infected browser to a mobile-based system via a challenge/response mechanism. The challenge and response generation process depends on validating the submitted information and ensuring the mobile phone legitimacy. Both phases within the MUAS context mitigate the denial-of-service (DOS) attack via registration information, which includes the client's mobile number and the International Mobile Equipment Identity (IMEI) of the client's mobile phone.

This novel authentication scheme circumvents the MITB attack by utilising the legitimate client's personal mobile phone as a detached platform to generate the challenge response and conduct business transactions. Although the MITB attacker may have taken over the challenge generation phase by failing to satisfy the required security properties, the response generation phase generates a secure response from the registered legitimate mobile phone by employing security attributes from both phases. Thus, the detached challenge- and response generation phases are logically linked.

To my Mother and Father

To my Brothers

To my Husband

&

To my Kids

Acknowledgements

First and foremost, I would like to thank God for giving me the strength and patience to carry out and finish this research. Second, I would like to express my sincere gratitude to my primary supervisor Dr. Imed Romdhani. I thank him for his ongoing guidance, encouragement, support and limitless patience and faith in me throughout this research. I would also like to thank my second supervisor Prof. Bill Buchanan for helping me with valuable suggestions and guidelines throughout my PhD. This work would not have been possible without either of them due to their enthusiasm, inspiration and their great effort to explain things clearly, not to mention their kindness and understanding in difficult times. I would also like to thank the kind help of Prof. Jessie Kennedy for being there for me during my study.

My thanks also go to Edinburgh Napier University and King Abdulaziz University for providing the appropriate environment for conducting this research.

I would also like to thank my friend and internal supervisor Dr. Etimad A. Fadel for being there for me throughout my studying period, advising and encouraging me and providing me with lots of valuable ideas. My friends and colleagues Dr. Omaimah O. Bamasag and Dr. Areej A. Malibary, I want to thank them for their guidance and help at difficult times during the thesis.

My deepest thanks go to my lovely husband Abdullah Baslaim, who supported and encouraged me in every possible way, and fed the confidence in myself and my ability to complete this work, when there were times that I felt it was impossible to finish. I would also love to thank my kids, Shahd, Omar and Sultan for coping with me and allowing me to finish my work.

Last in writing and first in my heart, my Mother and Father, whose prayers, support and continuous encouragement have helped and pushed me throughout the years to complete this work and never give up, even when things got hard.

Thank you.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Since the early days of e-commerce revolution, a world of cybercriminals emerged and produced a profitable online fraud business with a low risk of being caught [1]. An intermediate network has always been assumed as the point of compromise. Although this was true in the early days of networked computers, the internet evolution has allowed adversaries to live closer to and within end hosts [2]. Hence, a large amount of attention in academia and industry has been inspired to investigate more into the concept of developing a secure user-authentication system to secure the commercial transactions among the organization and individuals with respect to stopping host adversaries.

Most traditional e-commerce user-authentication systems involve the submission of confidential information, which can be prone to a number of attacks, such as a Trojan that infects the web browser, namely the man-in-the-browser (MITB) attack [1]. The strategy of this type of attack steals money from an online transaction [3] because of its ability to manipulate and intercept all incoming and outgoing information [1, 4]. Although the MITB attack is a serious issue, there has been an insufficient amount of research towards developing an MITB-prevention mechanism within the context of user authentication. Therefore, an investigation is conducted regarding the possible and available authentication methods to be used within the context of e-commerce applications to verify the identity of the client along with overcoming MITB attacks.

The current methods used for user authentication are categorised into single-factor authentication, two-factor authentication and three-factor authentication, which correspond to something the user knows, something the user has and something the user is, respectively [5-7]. The widely employed single-factor authentication is a simple password-based mechanism that employs a username/password combination to become authenticated. However, passwords can be easily stolen, forgotten or guessed [5], and this mechanism is vulnerable to various attacks [8]. In contrast, two-factor authentication is a much stronger mechanism that combines what the user knows with what the user has. Thus, it is based on the user’s possession of an object for providing a challenge/response mechanism to increase the level of authentication security [8-10]. The final authentication mechanism is based on authenticating the user via his/her biometrical characteristics[5-7].

According to Dougan and Curran [11], the only guaranteed solution to prevent MITB attack is via an out-of-band (OOB) confirmation, usually a one-time-password (OTP). On the other hand, Krishnan and Kumar [12] stated that the attacker can simply take control of the transaction following the OTP submission and user authentication. Therefore, MITB attack cannot be prevented with any of the previous authentication methods.

Given such a scenario, we argue that most proposed solutions utilise the browser for conducting business transactions, thus exposing the whole process to abort in case of an infected browser. We believe these solutions need to be modified to serve as an accountable and active countermeasure against MITB attack. In this thesis, the proposed authentication mechanism utilises the browser and the mobile phone as the initiation- and completion-phases, respectively. Where the detached second phase represents the solution to the MITB attack lying within the browser. A number of security-related technologies are combined to provide trustworthy mechanisms that logically link both phases, paving the way to generate a secure response within the detached and developed mobile phone application. The basis behind the modification relies on excluding the browser and continue the authentication mechanism securely within the application that can only be operated by its legitimate owner. The responsibility of the mobile phone application lies within capturing the displayed challenge, generating, and sending the response to the server without displaying any information on its screen.

Furthermore, the proposed scheme mitigate the connection-based distributed denial of service attack (DDoS) in both phases through exploiting the registration information within the server. This type of attack obstructs the legitimate user’s services through exhausting the server’s resources via application-level flooding attacks [13].

1.2 Objectives of the Research

Inspired by the lack of efficient solutions towards Man-In-the-Browser (MITB) attack and the desire to provide a secure authentication system within e-commerce environments, this study incorporated the required security measures along with the MITB capabilities to simulate its effect upon the proposed solution and to confirm whether the essential security requirements are satisfied.

This thesis provides a secure authentication system called "**Mobile User Authentication System**" or "**MUAS**", which is designed to utilise smartphones for a simple and secure client-authentication mechanism within the context of e-commerce applications for conducting business transactions. The MUAS is a two-phased system that uses a challenge/response mechanism for client authentication. Both phases are built on top of the secure socket layer/transport layer security (SSL/TLS) protocols as a mean for securing all connections and communications between participants. Furthermore, security measures are considered within both phases to prevent replay attacks and malicious users, thus satisfying security properties, namely authentication and secrecy.

This research focuses on three major objectives when developing a secure authentication system. **The first objective** concentrates on the main problem to overcome, which is the MITB attack. This involves defining its purpose along with its performance capabilities from an e-commerce application's perspective, which is explained in the following section. Based on MITB-attack capabilities, a secure user-authentication system is formulated that is composed of two physically detached, yet logically connected phases.

The second objective is to design and implement the two phases of the authentication system with a specific functionality for each phase. The second phase contributes to circumvent the problem of MITB attack by transitioning the authentication mechanism to the client's mobile device, which uses a detached connection for authentication and conducting business transactions. Entities involved in the two-phased system are characterised through their specific roles within the MUAS, where the client connects with the server through two separate and complementary roles. Furthermore, both phases mitigates the denial-of-service (DOS) attack through the client's registered information.

The third objective is to verify the two phases of the authentication system and to consider the effects of the threat residing in the browser. This part emphasises the need for requiring a detached second phase. Both phases are specified in an abstract manner, thus providing a detailed description of the protocols to be verified via the Scyther security-protocol-verification tool. The verification results are analysed per the security properties to be satisfied. Furthermore, this study is conducted in terms of computational and communication cost based on related studies and a theoretical performance measurement.

1.3 Man-In-The-Browser Attack

In this thesis, a type of current phishing attack is addressed, namely, man-in-the-browser (MITB) attack. MITB attack is a special case of man-in-the-middle (MITM) attack, where the attacker bypasses targeting the information flow in order to infiltrate and manipulate sensitive information provided by the client through attacking web browsers [4]. Thus, the attacker resides and acts in the application layer, within the open system interconnection (OSI) model that is the closest to the user. Hence, the attack occurs at the system level between the user and his/her browser [11]. From a structural point of view, a MITB attack is considered as a MITM attack between the user and the security mechanisms of the browser [11]. This type of attack is difficult to detect by the user or the server, as the attack might take place after the user becomes authenticated [12], where both parties receive deceptive information [1]. Thus, some standard authentication mechanisms employed within PCs, such as username/password combination, client certificates, secure ID tokens, one-time-passwords and biometric authentication, cannot be trusted to overcome MITB attack [14].

Moreover, MITB attack is invincible against the secure socket layer and transport layer security (SSL/TLS) protocols [1, 14], as the encryption mechanism is performed within the transport layer that is below the application layer, as illustrated in Figure 1.1. Thus, all exchanged data between the browser and the other party are exposed [1]. Whenever incoming encrypted data is received by the browser, it passes through the OSI model, starting from the physical layer and continuing up to the application layer, passing through the transport layer for decryption. As a result, the received encrypted data is disclosed. Conversely, outgoing data passes in plain text through the OSI model, starting from the infected application layer down to the physical layer, passing through the transport layer for encryption. Thus outgoing data is disclosed before being encrypted. In this scenario, incoming and outgoing data are disclosed to the MITB, even within a secure communication channel [1, 14].

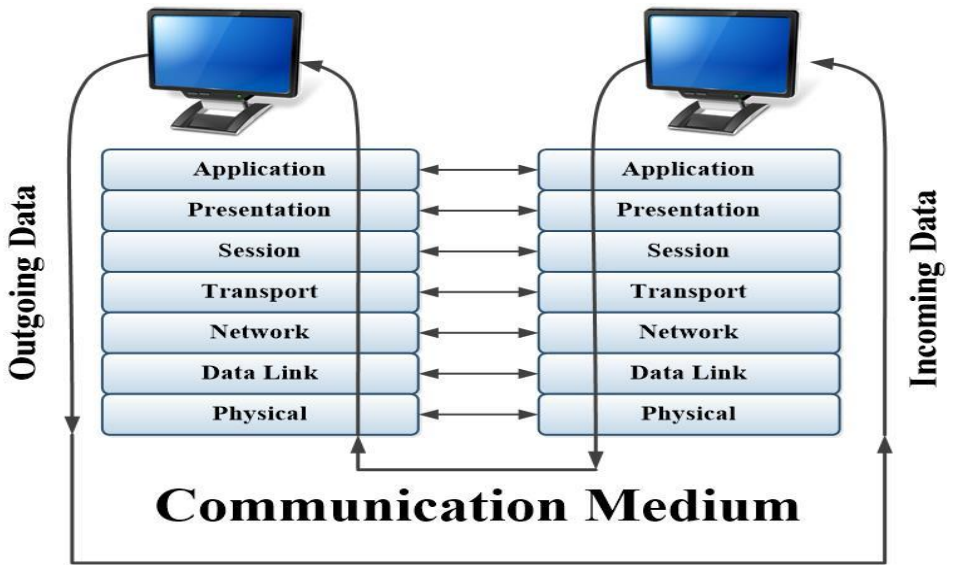


Figure 1.1: Data Flow through the OSI model

1.3.1MITB Attack Performance

Man-in-the-browser attack is a Trojan horse combined with phishing capabilities that infects a web browser to capture, manipulate or insert additional information to the web page without the user noticing [1, 14, 15].

Whenever a Trojan infects an operating system or a web browser application, it installs the malware extension into the browser configuration to be loaded and triggered once the user restarts the browser, thereby registering a handler for each page load. Whenever the user loads a specific web page, it is verified and filtered against a list of targeted sites. If the pattern verification matches, the user becomes authenticated via an authentication method and starts the transaction process. Once the user presses the submit button, the Trojan alters the user input before it is sent to the server, with neither the user noticing nor the server identifying the alteration. The alteration involves extracting user data via a document object model (DOM) interface to be modified and re-submitted to the server. The server generates the transaction receipt using the modified data to be sent to the browser. The received receipt will be detected by the extension, which will scan the receipt fields to re-modify the data with the original user data, in order to match the expectations of the user [1, 14]. Figure 1.2 shows the MITB behaviour within a banking system.

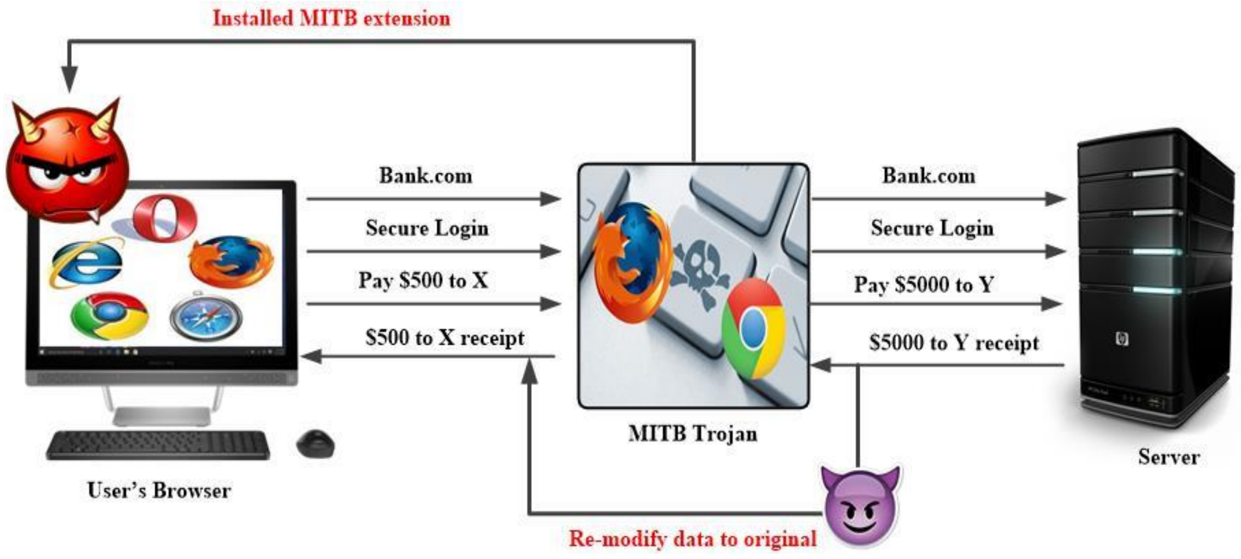


Figure 1.2: MITB attack within a banking system

In the case of an e-commerce application, the Trojan is able to capture the confidential information submitted by the user to be used in other fraudulent operations as illustrated in Figure 1.3.

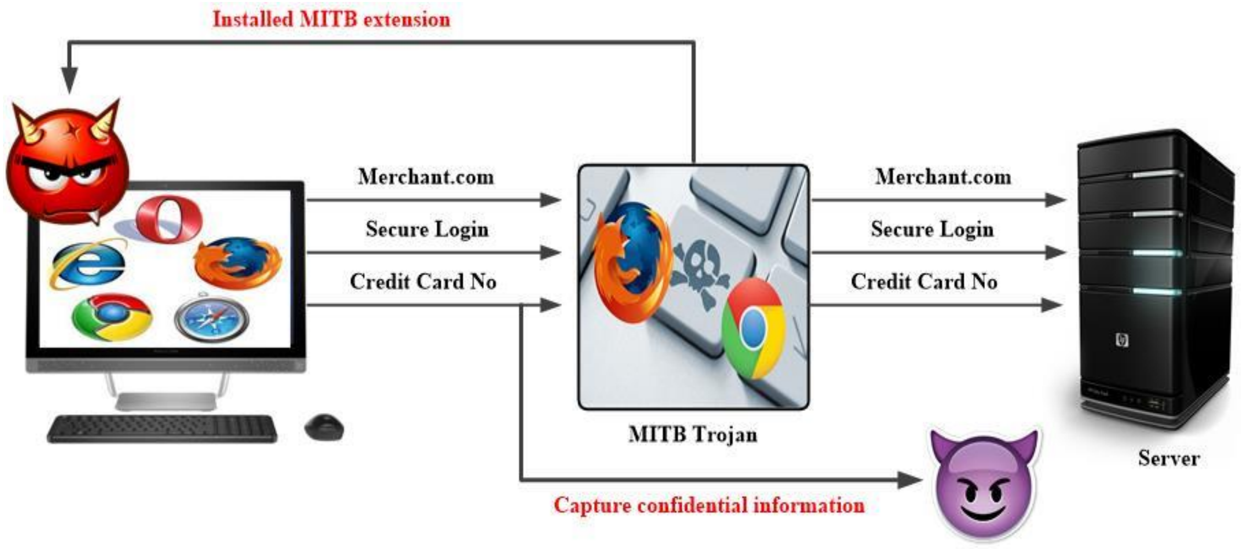


Figure 1.3: MITB attack within an e-commerce system

1.3.2MITB Attack Capabilities

In this section, the MITB attack capabilities are classified into four categories:

- **Stealing Data**: data submitted into a compromised browser can be passively and actively stolen by an MITB Trojan, along with the ability to choose the required data fields. Moreover, the MITB is capable of modifying the web page structure to prompt for more confidential information from the user, as most web sites today limit the amount of sensitive information required from the user. An example of this category involves stealing information within the context of e-commerce applications such as credit card numbers [11].

- **Modify HTML/HTML Injection**: this category refers to the ability of the MITB attacker to alter the HTML web page before being interpreted by the browser. Modifying HTML pages requires the Trojan to have a perfect background of the page format of the targeted sites, and it also requires a powerful data collection mechanism for obscure authentication methods which aim to defeat key loggers [11]. HTML injection can be performed in two ways:
    1. Adding extra data field within the web page, for the user to submit confidential information.
    2. Modifying the server response to reflect the original user input.

- **Modify Outgoing Data**: this usually happens within the context of online banking, where the Trojan is capable of modifying and tampering with user data without the user or the server discovering. This capability is very hard to detect, and it is unlikely to be discovered early which gives the attacker the opportunity to take advantage of it [11].

- **Choosing Targets**: MITB Trojan is capable of making a list of targeted sites to be available to its browser monitoring service, where two different types of data are selected from this list, either individual data fields of interest or the entire web page of interest. For each targeted site within the list, successful Trojans should be specific with respect to knowing what fields to inject and where, in order to carry out their attack. Therefore, MITB Trojans exclude non-valuable data in the process of stealing data and sending it back to the owner of the attacker [11].

1.4 Research Contributions

In order to conduct a secure business transaction within an e-commerce system, , this thesis made the following contributions:

- ***Designing a user-friendly and non-confidential input-based authentication system that bypasses the Man-In-The-Browser (MITB) attack via a two-phased authentication mechanism***

  The proposed MUAS requires the client to submit minimal and non-confidential information via a browser, which contributes to bypassing the MITB residing within the browser and exposing worthless information to the attacker. Upon submission, the proposed solution generates a secure challenge within the challenge-generation phase, in order to switch the authentication mechanism from the infected browser to a mobile-based system for conducting business transactions. This is achieved by utilizing the legitimate mobile phone of the registered client as its primary authentication device. The mobile-based system represents the response-generation phase that is physically detached, yet logically connected to the first phase via cryptographic security attributes and registration information.

- ***Mitigate the Denial-of-Service (DOS) attack within both phases of the MUAS via registration information***

  Upon mobile number submission within the first phase, the authentication server validates the received information against its registered clients. If the mobile number is unregistered, the server rejects the client and aborts the authentication process. Similarly, the International Mobile Equipment Identity (IMEI) extracted from the mobile phone within the second phase overcomes the DOS attack through validating the received IMEI against the registered IMEI to carry on with the client verification process. Thus, the proposed solution builds a strong authentication system against denial-of-service attack in both phases.

- ***Generating unique and useful cryptographic nonce-based QR-code contents***

The QR-code challenge (Quick Response Code) that is generated within the first phase represents the foundation for logically connecting the detached phases. For each session, the contents of the QR code are encrypted via public key cryptography (RSA), thus generating a one-time private and secure QR code. The contents of the challenge can be considered as unique and useful for being the basis of a one-time generated challenge and the basis for establishing a secure SSL/TLS communication channel.

- ***Exploiting security attributes from challenge- and response generation phases for generating the response, and overcoming the problem of a contaminated QR code***

  With respect to developing a secure protocol in both phases, security attributes are utilised to satisfy the mandatory security properties. One of these is the cryptographic fresh nonce, which is generated and exchanged between participants to prevent attacks. In addition to securing protocols, the employed cryptographic-security attributes represent the indispensable components required for generating the response, and contribute to linking the detached phases through a combination of the employed security attributes, thus forming the correct response within the mobile-phone application. Furthermore, the security attributes combination from both phases contribute to overcoming the problem of a contaminated QR code, where the attacker cannot exploit the recovered security attribute to generate a response from any mobile phone other than the phone of the registered client, due to its IMEI registration.

- ***Designing a three-level secure mobile-phone application***

  “3LS-Authenticate” is a user-friendly mobile-phone application that is responsible for capturing the QR code and generating the response. The application is composed of three levels, with each level meeting a security target. Although the encrypted QR code can be scanned by any QR code-scanning application, the mobile-phone application is the only application that can decrypt its contents.

  The application contributes to establishing a secure path with the server, along with exchanging a cryptographic nonce for satisfying security properties via exploiting the recovered-QR code’s useful content. While the other recovered-QR code unique content is exploited to generate the response through combining and hashing it along with the mobile phone's fresh nonce from the second phase, in the case of stolen mobile phones, the application is secure by only authorising legitimate phone holders to use the application.

- ***Applicability of the proposed authentication approach within other disciplines***

A minor modification within the QR code contents contributes in the capability of applying the proposed approach in other disciplines within the context of client authentication. Such disciplines include healthcare, academic institutions, or government applications, where all registered clients have a secret account number, for accessing their personal records within the server's institution. This account number will be contained within the QR code contents, for verification against the submitted account number within the mobile phone application, thus generating the response. Upon successful authentication, the client will be able to access their personal records.

1.5 Related Publications

The following publications have resulted from the research presented in this thesis:

[1] Molla, R., Romdhani, I., Buchanan, W. (2017). A Two-phase Mobile User Authentication System (MUAS) to Overcome Man-in-the-browser Attack. International Journal of Computer Networks & Communications (IJCNC)

[2] Molla, R., Romdhani, I., Buchanan, W. (2016). 3LS-Authenticate: an e-Commerce Challenge-Response Mobile Application. In: *13th ACS/IEEE International Conference on Computer Systems and Applications*. Agadir, Morocco.

[3] Molla, R., Romdhani, I., Buchanan, W., Fadel, Etimad Y. (2014). Mobile User Authentication System for E-commerce Applications. In: *International Conference on Advanced Networking, Distributed Systems and Applications*. Algeria

1.6 Structure of the Thesis

In **chapter 2**, preliminaries of the proposed system are introduced, where different authentication mechanisms are presented along with a detailed description of the technologies employed within the proposed Mobile-User-Authentication System (MUAS). Related work is also presented with regard to MITB attack.

In **chapter 3**, the infrastructure of the proposed MUAS is presented. This chapter details and justifies every aspect within the design phase along with objectives and security requirements of both phases.

In **chapter 4**, an implementation outline of the proposed MUAS is presented. This involves presenting the main algorithms of both phases in addition to the MUAS SSL-server platform establishment process, which is the essence of the authentication system. Moreover, components constituting both phases are presented along with their procedures.

In **chapter 5**, the verification and security analysis of the secrecy and authentication security properties are presented. This chapter provides a detailed description of the protocol through performing a theoretical performance measurement to evaluate its security with respect to possible attacks. Moreover, an implementation evaluation is conducted with respect to the most related work. The results of introducing the effect of the MITB attack into the first phase are presented along with the results of verifying the second detached phase.

In **chapter 6**, a summarization of the thesis is provided. This chapter concludes the findings from this thesis. A discussion regarding future work is also provided.

2 Literature Review

2.1 Introduction

Developing a secure mobile user-authentication system encompasses a wide range of stages. The first stage revolves around introducing the platform of this chapter, namely the e-commerce security system. This system is composed of several layers with associated security technologies that help to guide the work throughout this thesis.

The organization of this chapter is as follows. Section 2.2 describes the basic building blocks of this chapter by defining each component of an e-commerce security system, their associated technologies, and the required e-commerce security dimensions. Section 2.3 discusses the primary authentication methodologies along with their drawbacks and attack vulnerabilities. Furthermore, a review of existing UK bank's authentication mechanisms is accomplished to demonstrate the qualities of the most popular authentication mechanisms. Section 2.4 investigates the current trends in online authentication methods, techniques and technologies. The thesis objective is satisfied by leveraging these trends to develop the proposed MUAS. Section 2.5 discusses the technologies associated within the encryption technology layer of the e-commerce security system, namely symmetric and asymmetric cryptography, along with its most popular data-processing algorithms. Section 2.6 presents some of the authentication technologies employed within the encryption technology layer. Section 2.7 discusses and justifies the use of a widely-used security protocol within the security protocol layer that establishes a secure communication channel between participants. Section 2.8 reviews related work within the context of preventing MITB attacks and exploiting QR codes within authentication systems. Section 2.9 concludes the findings of the literature review chapter.

2.2 E-commerce Security

The objectives of e-commerce security are varied and include securing e-commerce assets from unauthorised access, modification, disclosure or use, thus protecting the confidentiality and privacy of the end user [16]. One of the main reasons for the loss of consumer trust in e-commerce applications is the disclosure of confidential information during a transaction. This is particularly true when the resulting data breach is due to a lack of security measures. Therefore, security is considered as the most important concern that needs to be addressed to guarantee client trust and e-commerce success [17].

With the revolutionary growth of e-commerce applications, a comparable number of security threats have surfaced that require a security presence to protect transactions [18]. As a result, secure technologies are being developed and amended every day [19]. Mapping these technologies to the seven-layered open system interconnect (OSI) leads to better technology characterisation, thereby reflecting the functionality of each layer within the e-commerce security system [20].

Generally, e-commerce security is divided into two main categories: computer network security and e-commerce transaction security. Computer network security plays a pivotal role in IT systems. As most applications operate within a networking environment, they therefore rely on the network's reliability, performance, and security [21]. It follows that, computer network security represents the foundation for e-commerce transaction security, as it focuses on implementing security solutions to solve security problems within a computer network, such as firewalls, vulnerabity scan, content aware, viral prevention and control [20, 22]. From a design point of view, deployed security safeguards may not be capable of accomplishing their function properly due to design or configuration errors [21]. Moreover, basic TCP/IP networking protocols do not provide cryptography nor strong authentication, resulting in exchanging unencrypted data and lack of confidentiality. Higher level protocols such as Telnet and SMTP do not provide sufficient security measures outside the username and password mechanism [23]. The inability of the traditional networking protocols to enforce strong security measures within e-commerce transactions result in a lack of security. Therefore, e-commerce transaction security focuses on solving security problems associated with online businesses. It processes e-commerce transactions in a secure and smooth manner over the foundation of computer network security through employing: (1) a safe transaction protocol, (2) a secure authentication mechanism to ensure the authenticity of the participant's identities, and (3) an encryption technology to prevent unauthorized information leakage. Hence, both computer network security and e-commerce transaction security are indispensable, as they complement each other. Considering the security requirements for e-commerce transactions, the security system model is divided into five layers: the network service layer, encryption technology layer, security authentication layer, security protocol layer and application layer [20]. Figure 2.1 demonstrates the five layers and their associated technologies [20].

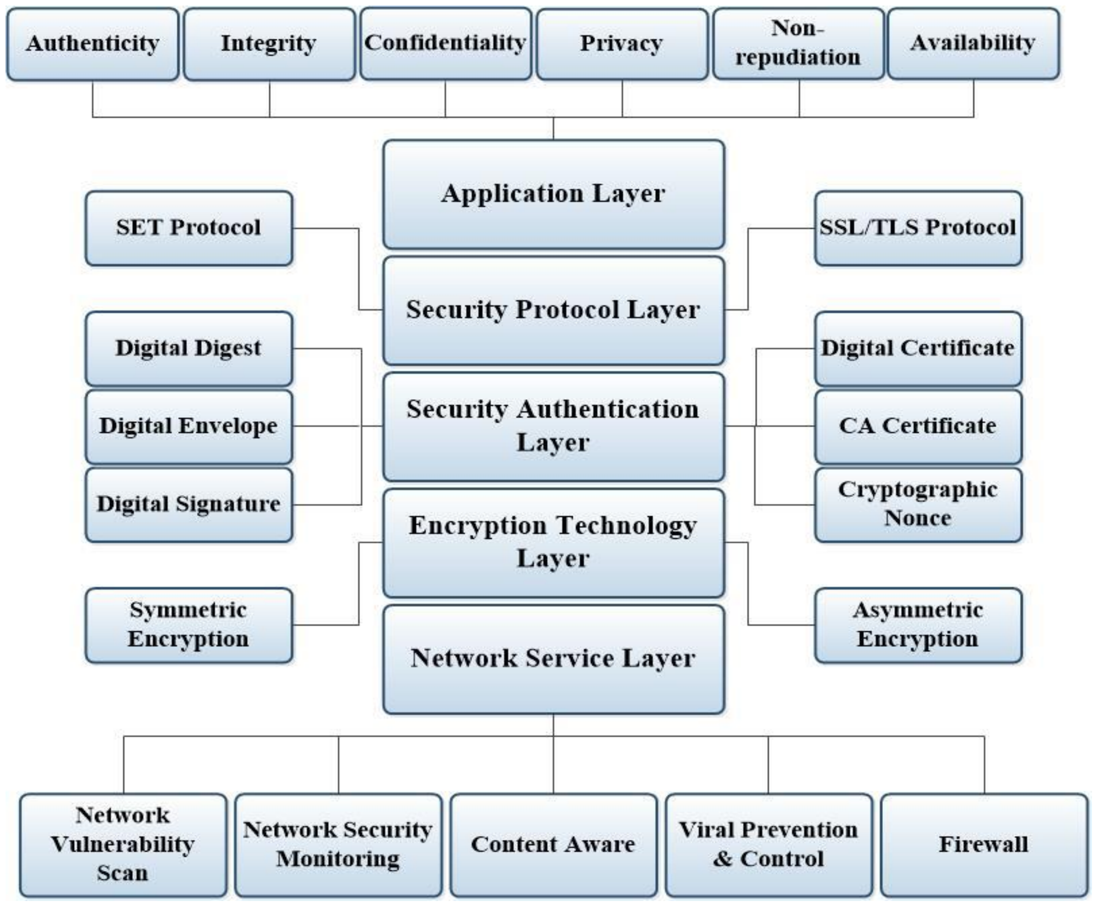


Figure 2.1: E-commerce Security System

Each layer is supplemented with example technology that serves the purpose of the layer as illustrated in the figure. In order to ensure the security of e-commerce transactions, e-commerce applications will be developed according to the open system interconnect layers (OSI), where all the essential e-commerce security layers will be placed within the OSI application layer. Each layer along with the technologies employed within the MUAS are presented in this chapter and are described according to the order of the layers.

2.2.1Dimensions of E-commerce Security

With the increasing growth of cybercrimes and cybercriminals, e-commerce security systems need to adopt additional measures to establish a secure e-commerce transaction. There are six key dimensions in e-commerce security: authenticity, integrity, confidentiality, non-repudiation, privacy and availability.

- **Authenticity**: the ability to verify the identity of a person or entity [24] and assure that the supplied identity is real and valid [20]. Authenticity is considered a prerequisite for allowing access to an information system [25].
- **Integrity**: the ability to guard proprietary data against tampering or alteration by an unauthorized entity while it is being transmitted over the network [24, 25]. Integrity can be achieved by message digest or hashing technologies [16].
- **Confidentiality**: the ability to guard personal data from being accessed by an unauthorized entity [24]. Moreover, confidentiality enforces authorized restrictions on personal data access via encryption/decryption, for personal privacy protection.
- **Non-repudiation**: the ability to ensure that e-commerce participants cannot deny their actions [24]. This means that the client receives proof of delivery and the merchant receives proof of the client's identity [25]. Non-repudiation can be achieved through the use of digital signature technology [16].
- **Privacy**: the ability to control the use of a client's personal information provided to the merchant [24].
- **Availability**: the ability of an e-commerce application to function as planned by preventing data delays or removal [24, 25].

2.3 Authentication

According to the National Institute of Standards and Technology (NIST), authentication is defined as the "process of establishing confidence in the identity of users or information systems" [26]. Another perspective by Cisco states that "authentication is the way a user is identified prior to being allowed access to the network and network services" [27].

In the e-commerce world, it is acknowledged that authentication is the first step towards establishing a secure connection with any organization. Therefore, both online merchants and clients need to acquire authentication information about one other as a means of identity verification [28].

All authentication methodologies can be broadly classified into three categories: single-factor authentication, two-factor authentication and three-factor authentication. These correspond to the following: something the user knows, something the user has and something the user is, respectively [5-7].

Single-factor authentication is based on the principle of "proof-by-knowledge" [8] via the use of static passwords [8, 29, 30]. Password-based authentication is the most common [6] and widely utilized mechanism for identity verification within communication systems [8, 29]. This mechanism is characterised by secrecy [5], simplicity and no requirement for additional hardware or software. On the other hand, passwords can often be forgotten, stolen or guessed [5]. Single-factor authentication is also considered to be a weak mechanism due to its vulnerability to attack [8, 31-33]. Although passwords are substantial, they do not provide users with a strong authentication mechanism [32] unless combined with other methods [6].

Two-factor authentication (2FA) is an authentication approach based on the "proof-by-possession" principle [8]. Here, the user possesses an object (e.g., hardware token or a mobile phone) to be combined with something the user memorizes (e.g., static password, PIN, pass phrase) [8, 9] to protect lost or stolen tokens [5]. This approach enhances the strengths of knowledge-based authentication by combining the two authentication factors, thus providing a challenge/response mechanism as a means of increasing authentication security [10]. The challenge/response mechanism is based upon generating a challenge on the server side that is to be processed by the client in order to produce the authentication response [8]. 2FA is a more secure mechanism than one-factor authentication, since a token can provide a strong defence against brute-force attacks [5]. However, it is still vulnerable to token attacks, man-in-the-middle attacks and session hijacking attacks [8].

Three-factor authentication is a strong authentication mechanism that is based on the "proof-by-property" principle [8]. In this approach, different user properties are used for remote identity authentication [34] using different biometric analyses such as fingerprint verification, iris analysis, facial analysis, voiceprint, handwritten signature verification and keystroke analysis [5-7].

Biometric-based authentication were considered a reliable authentication mechanism due to providing potential high-entropy information that cannot be lost or forgotten such as fingerprint, voiceprint, and iris scan [6, 7]. However, some biometric characteristics can be stolen from others, such as fingerprints that can be obtained from a mug [6] and built with silicone, gelatin, wood glue or latex [35]. Facial features can also be obtained from a photograph [35]. Therefore, not all biometric authentication approaches are trustworthy [6, 36]. Moreover, there is always the possibility of false rejection rate (FRR) and false acceptance rate (FAR) within a biometric authentication system. The FRR is the probability measurement that the system will falsely reject an access attempt by an authorized user. Whereas the FAR is the probability measurement that the system will falsely accept an access attempt by an unauthorized user. This type of inaccuracy is related to the lack of security requirements [35]. Security requirements of any biometric system should include confidentiality, integrity, availability, and authenticity. The confidentiality and integrity requirements should be satisfied for the acquired biometric raw data and the user template. Whereas the availability requirement should be satisfied for the biometric sensor, transmission channels connecting the different components of the target system, user template, and the processing data function implemented on the feature extractor component. Finally, the authenticity of the system decision is linked directly to the biometric system's functionality [35].

Furthermore, there exist other general issues that obstruct the biometric system's performance such as: sensor quality that is affected by noise, non-universality that leads to Failure to Enrol, lack of individuality that increase FAR, susceptibility to circumvention that leads to unauthorized access and, finally, intra-class variation and intra-class similarities that leads to increase FRR and FAR, respectively [37].

Table 2.1 lists all of the authentication methodologies along with their characteristics [5].

Table 2.1: Attributes of Different Authentication Methodologies

	User Authentication		
	Single-factor authentication (Knowledge-Based)	**Two-factor authentication (Object-Based)**	**Three-factor authentication(ID-Based)**
Commonly Referred to as	Password/ Secret	Token/ Mobile Phone	Biometrics
Support Authentication by	Secrecy	Possession	Uniqueness and personalisation
Security Defence	Closely kept	Closely held	Forge-resistant
Security Drawback	Less secret with time	Insecure if lost	Difficult to replace

With regard to online attacks, Schneier [38] introduced the “Attack Tree” method, which describes and analyses different types of attacks against system security [8, 39]. This comprehensive attack analysis provides a foundation for and facilitates the study of the existing authentication mechanisms [8], particularly the single-factor, two-factor, and three-factor authentication methods. Table 2.2 shows the applicability of different attacks on the three main authentication mechanisms [8].

Table 2.2: Applicability of Attacks to the Main Authentication Mechanisms

Authentication Methods / Attacks	**Static password (Single-factor authentication)**	**Challenge/Response (Two-factor authentication)**	**Biometrics (Three-factor authentication)**
User Surveillance	A	X	X
Token/note Theft	A	X	X
Hidden Code	A	X	A
Worms	A	X	A
E-mails with Malicious Code	A	X	A
Social Engineering	A	X	A
Webpage Obfuscation	A	X	A

Pharming	A	A	A
Sniffing	A	A	A
Active Man-In-The-Middle Attack	A	A	A
Session Hijacking	A	A	A
Brute-force Attack	A	X	A
Security Policy Violation	A	A	A
Website Manipulation	A	X	A
Man-in-the-Browser	A	A	A

Where:

A: Applicable

X: Not Applicable

Table 2.2 shows that two-factor authentication is prone to less attacks than static passwords and biometrics. Therefore, 2FA is considered as the most suitable authentication mechanism. However, it is still prone to the MITB attack, but one of its best and most popular features can be utilized within the context of our thesis, namely using the mobile phone as the primary authentication device.

2.3.1Password-Based Authentication

In 1981, Lamport [40] was the first to propose the concept of password-based authentication. The concept is based on authenticating a user based on an inputted username-password combination. Here, the system only stores the password-hash result to be compared with the submitted password after applying the hash function [32, 40]. Such a mechanism, however, is vulnerable to password theft. Nevertheless, it is still utilized to access some internet applications that doesn't require strong protection against repudiation [5], such as e-mail services and social networks [6, 8], although its security properties [41] are not reliable [6]. Requesting longer and more complex passwords in an attempt to secure this mechanism has led users to write down their passwords, making them prone to theft [8].

Although password-based authentication takes place via a secure socket layer (SSL) channel [42], it is still vulnerable to sniffing attacks [8, 31] within the SSL handshake establishment process [43]. Here, the sniffer masquerades as the server and provides the user with a fraudulent certificate [8]. Therefore, the system also becomes vulnerable to man-in-the-middle (MITM) and phishing attacks [8].

Alongside password-based authentication vulnerabilities [8], users are required to manage and memorize too many passwords [44]. Therefore, specialized password-manager (PM) software is available [44], which “manages and organizes passwords and personal identification in a secure manner” [45]. Such software can help reduce the effort in creating, memorizing, and changing passwords [45]. Password managers are classified into web-based and browser-based PMs. However, while password managers are secure against phishing and pharming, they are still vulnerable to failures, which can be due to the web-based PM server being targeted by cyberattacks, or the browser-based PM information being stolen or lost due to damage to the local hard disk [45]. Furthermore, a security analysis conducted on five popular web-based PMs, found that four of them—LastPass, RoboForm, My1Login, and PasswordBox—have critical vulnerabilities in which attacker can obtain the user’s credentials [46]. A good remedy for this is to include an extra object, which is possessed by the user and entered into the authentication mechanism, thus forming a stronger two-factor authentication [6]. The password-based authentication mechanism cannot be employed as a secure stand-alone system to be used within financial institutions and e-commerce.

2.3.2Two-Factor Authentication

To increase security and overcome the problem of stolen passwords in the password-based authentication mechanism, a server-side countermeasure that includes adding a second authentication factor is frequently employed by online services [44, 47, 48]. This countermeasure uses an out-of-band (OOB) communication channel as a means for receiving a one-time-generated token to be used alongside the standard password-based authentication mechanism [44].

In 1991, Chang and Wu [49] proposed the idea of a two-factor authentication using passwords and smart cards as a means of client authentication [6]. Since then, the trend towards proposing two-factor authentication solutions has increased [6], with the use of other objects, such as smart cards, security tokens, and mobile phones, as the second factor within the authentication scheme [50].

Over the past decade, a growing number of banks have started to support online banking as a means to provide customers with financial services anywhere and anytime [47]. Obviously, a stronger and more secure authentication mechanism was required [47]. Therefore, online banking sites switched to the two-factor authentication method as a simple and friendly way to securely authenticate online banking end-users [47, 51]. In a study by Krol et al. [47], looking at the identification and authentication mechanisms used by 11 popular UK banks, all banks were found to follow the two-factor authentication mechanism for logging into their online banking platform. The first authentication factor is usually based on the user's knowledge, such as a username-password combination, pass phrase, or PIN number, while the second factor can be based on challenge questions on memorable information [47], a security token generator issued by the bank [47, 51], a mobile banking application that generates random numbers [47, 51], or a generated random number sent to the customer through SMS [47, 51]. When reviewing these popular banks' official websites, it was found that most of these banks employ both one-factor and two-factor authentication mechanisms. The one-factor mechanism is used to provide limited banking services, such as viewing account balances, transactions, and checking and savings account statements using e-Statements [52-54]. In addition, customers can make payments to existing payees, transfer money between accounts in the same bank, and report lost or stolen cards [52-54]. The two-factor authentication mechanism for login through security tokens or mobile banking applications gives customers access to the same services as with the single-factor authentication, plus some additional ones, such as making payments to new payee, making international payments, changing personal details, and managing regular payments [52-54]. Using the banks' own terminology, Table 2.3 summarizes the different identification and authentication mechanisms employed by different UK banks.

Table 2.3: Different Identification and Authentication Mechanisms Used in UK Banks

	Identification Mechanism	**Authentication Mechanism**	
		First Factor	**Second Factor**
Barclay's [53, 54]	Surname + Membership Number, Card Number, or Account Number	1) 5-digit Passcode 2) 2 out of the 8+ characters of memorable word	-

		8-digit OTP (PINsentry)	
HSBC [52, 55]	Username	1) Memorable Answer 2) Password	-
		Memorable Answer	6-digit OTP (Secure Key)
First Direct [56, 57]	Username	Memorable Answer	6-digit OTP (Digital Secure Key)
Nationwide [58, 59]	Customer Number	1) Memorable data 2) 3 out of the 6-digit pass number	-
		8-difit OTP (Card Reader)	
Halifax [60], Lloyds [61]	User ID	8+ Character Password	3 out of the 6+ Character Memorable Information
NatWest [62, 63], Ulster [64, 65]	Customer Number	1) 3 out of the 4-digit PIN 2) 3 out of the 6+ Character Password	-
		8-digit OTP (Card Reader)	
RBS [66]	Customer Number (DOB + 4 digits)	1) 3 out of the 4-digit PIN 2) 3 out of the 6+ Character Password	-
Santander [67]	Personal ID	1) 8-digit Password 2) 5-digit Security Number	-
		SMS-OTP	
The Co-operative Bank [68]	Account Number or Credit Card Number	1) 2 out of the 4-digit Security Code	-

		2) Memorable Information	
Standard Chartered [69]	Username	SMS-OTP	
Bank of Scotland [70]	User ID	Password	-
		Bank Secure App / Phone call (Keying a displayed 4-digit [OTP] into the phone number pad)	

In terms of security, Tables 2.2 and 2.3 show that two-factor authentication has been adopted by most financial institutions, for whom security is the number one concern. However, the traditional 2FA cannot be useful within the context of our thesis, due to its vulnerability to the invincible MITB attack. Therefore, the essence of the mechanism can be utilized through employing the mobile phone within the authentication framework to replace the presumably infected browser.

2.3.3Three-Factor Authentication

As an improvement over some of the security issues with two-factor authentication, Juels and Wattenberg [71] proposed the first three-factor authentication scheme, which uses biometric identification. According to the National Science and Technology Council (NSTC) subcommittee on biometric and identity management [72], biometrics are defined as measurable biological (anatomical and physiological) and behavioural characteristics that can be used for automated recognition [73].

Biometrics provide the “something you are” factor of authentication to strongly authenticate legitimate users and detect fraudsters through physiological and behavioural characteristics [74]. Recognition of physiological characteristics, often known as “what you are” [75], includes human body measurements such as fingerprint recognition, facial recognition, retina recognition, iris recognition, and hand recognition [74]. On the other hand, recognition of behavioural characteristics, known as “what you do” [75], includes measurement of human action, such as voice recognition, signature recognition, gait patterns, and key stroke dynamics [74].

Biometrics substitute for the “proof by knowledge” and “proof by possession” mechanisms [8], since passwords can be easily forgotten/stolen and devices can be lost/stolen, respectively [74].

Common reasons to choose biometric authentication over password-based authentication are based on:

- Physical presence: Biometric authentication requires the person being identified to be present at the time of authentication, which makes this type of authentication more secure [76].
- No need to remember information: The biometric information required for authentication is always present, which bypasses the need to remember any sort of information [76].
- Less prone to forgery: The possibility of forging or faking biometric identity is much lower than in password-based authentication [76]. However, it is not impossible [75].

To improve security in a password-based authentication system, biometrics can be incorporated into the scheme [76]. As a solution, Bio-key International, Inc. [77] introduced a mini USB fingerprint reader called “SideSwipe” [78] and a compact fingerprint reader called “EcoID” [79], which are used as alternatives to passwords, tokens and PINs. SideSwipe [78] and EcoID [79] are multi-factor authentication solutions that can be used for healthcare, banking, retail, identity and access management. Most Bio-key International products operate on Microsoft Windows devices.

2.3.3.1 Biometric Properties and Drawbacks

A biometric system must satisfy a number of properties in order to be a successful and suitable authentication system. The following properties were proposed by Clarke [80] after analysing biometric system requirements.

1. **Universality**: The chosen biometric characteristic within any authentication system should be owned by each person, and it should be difficult, if not impossible, to lose [75, 76, 80].

2. **Uniqueness**: Biometric characteristics within authentication systems must distinguish between individuals, such that no two persons would ever have the same biometric values [75, 76, 80].
3. **Permanence**: Long-lasting biometrics that do not change over time due to age or disease should be used within authentication systems [75, 76, 80].
4. **Collectability**: The chosen biometric characteristic must be easily collected from anyone on any occasion [75, 76, 80].
5. **Acceptability**: The chosen biometric to be collected must be acceptable to society [76, 80].
6. **Measurability**: The digital device/sensor used to acquire and digitize the biometric characteristic should be suitable, causing no inconvenience to the people [76, 80].
7. **Circumvention**: In the case of forged or faked biometric characteristics, the authentication system should have the ability to handle such situations [76, 80].

Finding a biometric characteristic that satisfies all of the above properties is difficult due to drawbacks regarding some biometric characteristics, biometric readers, template storage, and costs [75].

Fingerprints do not satisfy the universality property, since some individuals do not have clear fingerprints due to aging or injury [81]. Some cancer patients can lose their fingerprints due to some kind of chemotherapy [81]. While facial recognition is much more universal, its error rate is much higher due to its poor uniqueness property [75]. In some cases, multi-model biometric systems combine several biometric characteristics to overcome some of the drawbacks of using uni-model biometrics [82]. For example, combining facial and fingerprint recognition increases the recognition rate and decreases the error rate [83]. Another issue regards the biometric reader acceptability, as in the case of retina reader. In an outrageous form of attack, biometric collection is performed by the attackers, who record the biometric characteristics without the owner’s knowledge [75]. Furthermore, some biometrics can be copied, in the case of fingerprints [84], or forged in the case of the iris [85] by employing non-invasive biometric readers. The more invasive a reader is, the harder it is to copy a biometric [75]. This leads us to a related characteristic, which is cost. The less expensive the biometric reader is, the higher the likelihood of it being fooled by forged characteristics, while more expensive fingerprint readers with wide functionality can check that the fingerprint is being inputted by a live person [75]. Finally, the storage systems that hold the biometric templates must be highly secured and protected [75].

2.4 Current Trends in Online Authentication Solutions

A large number of studies have been conducted with regard to developing online authentication solutions. In this section, two of the most popular technologies, namely one-time passwords and QR codes are discussed in detail along with their related work, attacks and vulnerabilities.

2.4.1One-Time Password (OTP)

Among all second factors, the system's capability to dynamically generate a login-session password provides the best enhancement of security within two-factor authentication systems. The dynamic login-session password is called a one-time password (OTP) [86], and it is considered one of the strongest authentication mechanisms against account theft [87, 88]. A one-time password can be defined as a temporary password that is valid for one use only [89] within a single session or transaction [9].

The basic idea behind the development of OTP, first suggested in 1981 by Lamport [40], was to overcome the security concerns accompanying static passwords and to avoid man-in-the-middle (MITM) attacks [40]. The scheme was based on the generation of different pseudorandom outputs each time the user tries to login [9, 40]. Successive tests of OTP in many authentication systems alongside different technologies have been conducted in different areas such as online banking [44, 90-92], healthcare [91, 93, 94], academic institutions [95], and government applications [91]. According to a two-factor authentication usability study by Krol et al, [47], in the context of online banking within the UK, it was found that many banks require the OTP authentication method. Banks rely on hardware tokens as a means to generate the second authentication factor. Such devices can be either a secure key such as the HSBC PIN-protected security device [55], or a card reader that requires a debit card to be inserted into the device, such as Barclay's PINsentry security device [96] and NatWest's card reader [97].

In an attempt to develop user friendly and secure online authentication mechanisms, researchers have conducted a large number of studies on the use of OTP with different authentication technologies. Some of these studies are presented in this section. For instance, Tandon et al. [92] proposed the integration of OTP with QR code technology in a novel authentication scheme for online banking. The scheme involves sending the client an e-mail with a QR code containing an encrypted OTP, to be verified through the mobile banking application by entering the ATM pin to decrypt the OTP. Although the system makes it difficult for intruders to detect secure information, user friendly that requires no technical pre-requisites, and provides authenticity, integrity, confidentiality, and privacy, it is still vulnerable to man-in-the-browser (MITB) attacks, in which all computations take place within the browser. A similar methodology based on embedding an OTP into a QR code was proposed by Moholkar et al. [98] to detect phishing websites. The proposed anti-phishing framework verifies the originality of a particular merchant's website as a means to protect personal information and passwords from phishing websites. The bank possesses a database of registered original websites, and another database for their user accounts. The framework login phase begin with the user submitting his or her credentials to log into any merchant website. The merchant sever sends this information to the bank, along with information about the site, in order to generate and embed an OTP into a QR code, which is split into two parts. The first part is sent to the user through e-mail, while the second is sent to the merchant server through network and then sent to the user via this network. The user combines both parts to scan the QR code and recover the OTP, which is sent to the bank through e-mail for verification. Finally, the bank sends an e-mail to the user, stating whether the website is phishing or not. The system protects online users from attackers by dividing the image into two parts and sending them to different servers, since one part does not reveal the entire image. On the other hand, the system cannot be considered user friendly due to the continuous need for e-mail use. An additional factor is the overhead caused by the computation required to combine the two images.

A new approach for securing ATM authentication systems using fingerprint identification and OTP was proposed by Shamdasani and Matte [99]. The approach works by verifying a currently scanned fingerprint against a biometric template that was registered during account opening. If the fingerprints are a match, the bank sends a random four-digit OTP to the client’s mobile phone to be submitted by the user for authentication purposes. Compared to the traditional magnetic card reader used by the majority of ATM machines, the proposed approach provides a safer authentication mechanism. The only disadvantages of this system are the shortcomings associated with biometric systems, such as implementation cost, damaged fingerprints, and people’s concerns with touching a public device.

2.4.1.1 OTP Attack Methods

With the evolution of attack methods, hackers have developed MIT-X (man-in-the-X) attacks, which combined with other concepts to attack accounts under OTP protection. MIT-X attack methods include:

- Man-In-The-Middle (MITM)
- Man-In-The-Browser (MITB)
- Man-In-The-PC (MITPC)

An MITM attack combined with phishing is used for to attack OTP. The attacker first attacks the user’s PC to intercept the certification and snatch credentials through a phishing website. Next, when the user submits the OTP to log in, the attacker intercepts the OTP value through an MITM attack. The second attack is an MITB attack (man-in-the-browser attack), caused by malignant code installed in the web browser. The main idea behind this type of OTP attack is to use a malignant program to cover over the targeted internet banking service with a fake input form, without the need to create a fake site. Finally, there is the MITPC attack (man-in-the-PC attack), which exploits PC environment vulnerabilities to control and attack every data access pathway. This includes keyboard and mouse logging, screen capture, and accessing memory to intercept fundamental account information from the user’s PC. Furthermore, a new type of attack that combines MITPC and MITM has emerged. This method uses an MITM attack to intercept the OTP, which will be used with the information previously collected during the MITPC attack for account theft [88].

2.4.1.2 OTP Distribution Mechanisms

There are a number of ways in which one-time passwords are generated or distributed [88], such as through SMS messages [8], mobile banking applications [47] and hardware tokens, which are small devices given to the client to aid in the authentication mechanism. Although tokens provide a secure environment, they can be very costly for organizations in terms of customer training and replacing lost or damaged tokens. There is also the burden of having to always carry and maintain the token. This could be especially difficult if the customer has several accounts at several banks, which would lead to the inconvenience of having more than one token. Therefore, this subsection will only focus on SMS-OTP and mobile banking applications.

2.4.1.2.1 SMS-based One-Time Password (SMS-OTP)

The SMS-based channel has been the predominant OTP distribution mechanism for online banking in a number of countries, including Germany, Spain, Switzerland, Austria, Poland, Holland, Hungary, the US, China [48], and Saudi Arabia. Moreover, a number of online service providers such as Google, Facebook, Dropbox, Microsoft, and Apple have recently been employing SMS-OTP as the second authentication mechanism in their login verification systems [48].

In SMS-based one-time-password verification systems, users log in with their standard username-password combination [90], followed by submitting the verification code (OTP) received on their mobile-phone via SMS [90]. SMS-OTP was built upon two foundations: cellular networks and mobile phones. Today, these foundations are quite different than what they were when SMS-OTP was designed [100]. An attacker’s main goal is to get a hold of the OTP, which can be done either through wireless interception or mobile phone Trojans [100]. In terms of wireless interception, GSM technology is insecure due to a number of vulnerabilities, such as unilateral authentication, which jeopardizes the network to MITM attack [100, 101], implementation algorithm flaws causing weak cryptographic algorithms [100, 101], and disabled wireless encryption in some countries or situations [100, 101]. As a result, over-the-air communication can be vulnerable to eavesdroppers and sniffers, who are capable of intercepting mobile traffic in order to get a hold of the authentication information [100, 101]. Furthermore, mobile phone Trojans have been created by criminals as a means of stealing money [100]. SMS-OTP-stealing Trojans are designed specifically to receive, alter, delete, and forward SMS messages that contains OTPs without leaving any trace [100].

Mulliner et al. [100] proposed two countermeasures to diminish attacks against SMS-OTP. The first approach uses the concept of end-to-end encryption, where a permanent and private storage area for each application is used to store the received OTP. The concept is based on using the SMS channel to send an encrypted server-side OTP to the mobile phone. The mobile phone decrypts the OTP using a dedicated application, which stores that piece of data in the application's private storage [100]. Although an SMS-OTP Trojan can still access received SMS messages, it cannot access the user-specific key that is needed to decrypt the OTP. The only problem with this approach is key distribution. Another study involving prototyping attacks against the SMS-based TAN schemes of four large banks was conducted by Dmitrienko et al. [48]. The study involved investigating dual infection attacks against a scheme with a server-side generated OTP and direct OOB. In terms of attack implementation, the login credentials are stolen from the PC though an MITB attack, while mobile malware intercepts SMS messages through an MITM attack between the GSM modem and the mobile phone and hides them from the user. The result showed that malware has succeeded in intercepting SMS messages, stressing the need for further research within the context of secure mobile two-factor schemes. The second approach proposed by Mulliner et al. [100] is based on creating a virtual dedicated channel inside the mobile phone's operating system to protect specific SMS messages against local Trojan interception, redirecting them to a special OTP application called "OtpMessages," which blocks them from being read by other applications. The only difference between the "OtpMessages" and the default SMS application is that "OtpMessages" only receives messages containing OTPs due to the use of a knowledge-based filter that matches keywords against the message body. This approach seems effective against Trojan attacks. On the other hand, it requires operating system manufacture support, along with some support from service providers and cellular network operators.

Hamdare et al. [89] proposed a technique to secure SMS-OTP from MITM attacks by combining the received SMS-OTP with a pre-shared secret key, to be passed through an RSA algorithm in order to generate a transaction password on the mobile application side. This will be compared to a copy of the same transaction password that exists on the server side when the client submits his transaction password through the website.

Finally, Saxena and Chaudhari [102] proposed a successful, secure and efficient end-to-end protocol called “EasySMS” for securing SMSs between end users, whether they share the same Home Location Register or not. The protocol is based on transmitting symmetric keys to mobile users for secure message exchange after verifying both end users. The analysis of EasySMS showed that the protocol prevents SMS disclosure, replay attacks, MITM attacks, over-the-air modification, and impersonation attacks. The protocol also generates less communication and computational overhead, it utilizes bandwidth efficiently by reducing consumption to 51% and 31% and, reduces message exchange during the authentication process by 62% and 45% compared to the SMSsec and PK-SIM protocols. Furthermore, within the protocol modelling phase, the authentication server was assumed to work as the authentication centre (AuC), in which all symmetric keys that are shared between the authentication server and the mobile phones are stored. Therefore, the proposed protocol could be applied within any framework that requires OTP generation on the server side. Another solution by Liu and Zhu [103] eliminated the use of SMS messages by utilizing Bluetooth technology to create a connection between a computer and a smartphone. Thus, the remote server sends the OTP to the computer, which will be submitted to the smartphone via Bluetooth. This scheme is based on submitting the Bluetooth device address to the remote server during the registration phase. As SMS is transmitted over a telecommunication carrier, the sender will be charged with fees for each sent message. On the other hand, mobile instant messaging (MIM) is transmitted via the internet, which is free of cost [104]. Therefore, the use of public MIM services has become increasingly popular due to providing a private network communication between two users, in addition to providing a chat session between two or more users [105, 106]. In MIM, users are authenticated by logging into the IM network via submitting their usernames and passwords. The logging authentication is performed by exchanging hashes of the password. In the IM approach, expensive cryptographic algorithms such as RSA are avoided. Instead, cheaper cryptographic algorithms such as MD5 and SHA are used. In the authentication phase, a sniffer who monitors exchanged packets is able to determine who is logged into the IM server without knowing the password. The well-known hash algorithms used for hashing passwords within IM achieves this. These algorithms can be applied to the challenge sent to the server along with the hash result, for recovering the password. Thus, the system is vulnerable to dictionary attacks [106] and impersonation in the case of a stolen username and password [107]. This type of data theft is performed off the wires, where the attacker grabs TCP packets off the network. Another type involves stealing conversation logs that are created by almost all IM clients. Moreover, a profile of an individual can be built from registration information that is publicly available (data mining), thus resulting in identity theft [107]. In terms of spreading viruses, vulnerabilities in the IM software or Trojans deceiving users into executing a malicious code can help spread viruses quickly to the user's contact list [107]. In terms of the application speed, whenever the system becomes busy, message sending and receiving becomes slower. A delay of more than one second means that the conversation will not take place [107]. Considering the mentioned security drawbacks of the MIM services, a trade-off is made with respect to security in favour of the SMS.

2.4.1.2.2 Mobile Banking Application OTP

With the increasing popularity of mobile phones, there has been a growing trend towards mobile phone investment to help surpass desktop PCs in providing mobile banking [108]. As Ha et al. [109] have indicated, the benefits of mobile banking include 1) ubiquity, 2) immediacy, 3) localization, 4) instant connectivity, and 5) proactive functionality. Therefore, large banks provide mobile banking, through which their clients can conduct secure financial transactions such as balance inquiries, payments, and transfers [110, 111]. Even small banks and credit unions are considering mobile banking as a means to simplify banking activities for their clients [112]. However, mobile banking security is complicated and is considered a major concern for mobile banking clients due to the existence of different mobile devices and platforms [112]. Therefore, a number of security methods are used in mobile banking. These include 1) passwords, 2) encryption, 3) security questions, and 4) downloaded applications [112]. These methods help divide mobile banking into three categories: password, messaging, and applications [110]. The first defence mechanism is the screen lock, which is equivalent in strength to the password-based mechanism, thus providing very weak security [110]. The second category is messaging, which revolves around receiving an authentication code (OTP) through the SMS channel, to be submitted within the banking application, as is done when activating the HSBC digital security device via the HSBC mobile banking app [113]. Finally, a large number of UK banks provide clients with downloadable mobile banking applications [54, 114-125]. Mobile banking applications are used to conduct secure financial transactions with the bank. The application mechanism requires at least two pieces of information in order to activate the app—the username-password combination. Moreover, some applications require a second layer of authentication, one that is linked to the authentication device to ensure that the mobile phone is in the possession of the account's legitimate owner [110]. This authentication factor is a one-time password (OTP) that is generated by the banking application itself and is required for internet banking transactions, as in the HSBC mobile banking app [113, 114] and Barclay's mobile PINsentry banking app [54].

2.4.2QR Code

With increasing security concerns, there has been a strong trend in online security towards employing QR code technology as a means of assuring online users that the service they are using is secure and reliable. [126]. This approach has rapidly gained popularity and found widespread adoption within two-factor authentication schemes, where QR codes are applied as the client's second factor within secure online transactions.

QR ("Quick Response") code is a machine-readable [127], two-dimensional matrix barcode technology that decodes its contents at high speed [103, 128, 129]. It was first developed by the Japanese corporation Denso-Wave in 1994 for tracking parts in the vehicle manufacturing process [128, 130, 131]. In June 2000, QR codes were approved as an ISO international standard (ISO/IEC 18004) to be made publicly available to people regardless of applicability [132]. Recent QR code usage has since expanded beyond its initial purpose to cover a multitude of different purposes [128], ranging from marketing [133] and advertising [134, 135] to secure mobile payments [135-138].

2.4.2.1 QR Code Structure and Capacity

QR code capacity is influenced by several factors:

- **Version**: there are 40 different versions of QR codes. Each version differs in the number of modules, which are the black or white dots constituting the QR code. Version 1 consists of 21 x 21 modules, from which 133 are used for storing the encoded data, while version 40 is the largest QR code symbol, consisting of 177 x 177 modules and up to 23,648 data storage modules [139].

- **Error correction level**: the QR code error correction feature is based on the Reed-Solomon codes. This robust feature restores data even if the code is damaged or partially dirty [140-142]. There are four error correction levels, namely L (Low 7%), M (Medium 15%), Q (Quartile 25%), and H (High 30%) [140, 142]. Error correction level L is the most preferred level [142], since higher error correction levels improve error correction ability by increasing the area reserved for error correction codewords, thus decreasing the area reserved for storing the actual data [135]. Furthermore, based on the error correction feature, QR codes are readable from any direction at a stable, high speed. This is accomplished through the position detection patterns located at the symbol's three corners [129, 141]. Another useful feature is masking, which helps increase the image contrast, and helps the reader software decode the QR code [142].

- **Encoded data**: QR code capacity depends on the amount of data that needs to be included. Whenever the data amount increases, more modules are required for the QR code, resulting in a larger symbol [139]. All data types are supported by QR code, including numeric, alphanumeric, kana, kanji, hiragana, symbols, binary, and control codes [129, 135, 141]. QR codes are able to encode up to 7,089 characters in a single symbol [129, 141].

Each QR code consists of eight sections, each of which is necessary for the scanner's ability to decode the data [142]. Figure 2.1 shows the different sections of the QR code [127, 142].

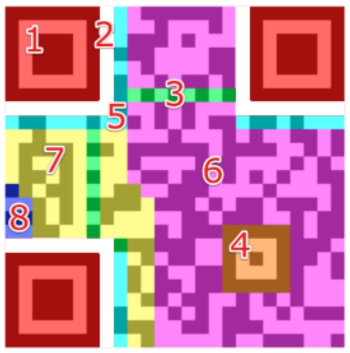


Figure 2.2: QR Code Structure (Version2)

Finder Pattern (1): The finder pattern is used to detect the QR code position and determine its correct orientation (omni-directional). It consists of three identical 3 x 3 square boxes (modules) located at all corners of the QR code, except for the bottom-right corner. Each pattern is represented by a square black box, surrounded by square white box, and surrounded again with a square black box [127, 142].

Separators (2): Separators are the white modules surrounding the finder pattern to improve its recognisability and to separate it from the actual data [127, 142].

Timing Patterns (3): Timing patterns are a sequence of black and white modules [142] placed in vertical and horizontal form. Each pattern has alternating dark and light modules, beginning and ending with dark modules. The vertical timing pattern extends across the sixth column, between the separators of the left-hand finder pattern, while the horizontal timing pattern extends across the sixth row, between the separators of the upper finder patterns. Timing patterns help the reader software determine the modules' width [127].

Alignment Patterns (4): Alignment patterns help correct possible code distortion when scanning the QR code. The symbol size determines the number of alignment patterns [127, 142].

Format Information (5): Format information consists of 15 bits next to the separators. It contains the error correction level and the selected mask pattern of the QR code [127, 142].

Data (6): The data section is where the data are stored in 8-bit pieces after being encoded into binary values called codewords [127, 142] and then converted into black and white modules. This section also contains version information, format information, and error correction codewords [127].

Error Correction (7): Error correction codewords are generated from the data codewords [142]. These error correction bits are stored in 8-bit-long codewords in the error correction section [127, 142].

Remainder Bits (8): Remainder bits consist of empty bits that are present in case the data or error correction bits cannot be divided into 8 bits without a remainder [127, 142].

Finally, surrounding the QR code symbol is a quite zone, which is a white margin that surrounds the symbol on all four sides. It consists of four wide modules that help improve symbol detection [127].

2.4.2.2 QR Code Usage

Recently, QR code usage has expanded and evolved significantly beyond its initial purpose [128]. QR codes are able to store different types of data, such as URL, contact information, e-mails, geo-location, and plaintext [142]. The marketing industry is widely employing QR codes as a means of advertising by encoding URLs, which take users to companies' official web pages without having to type an address [143].

A unique example of using QR code for advertising was done by a shampoo company, which introduced QR code haircuts. People were thus used as moving advertisements to be scanned, leading to the company's website [142]. Another clever and innovative use of QR codes was launched by Reporters Without Borders, an international and non-governmental organization whose goal is to defend freedom of the press [144]. The idea was to create a printout that literally speaks to the people; by scanning a QR code in a magazine and placing the smartphone on a published full page photo, audio of journalist speaking about what's really happening in the world would start playing [145, 146]. Furthermore, QR codes have been adopted by some companies to enable customers to purchase products or pay for services [142]. QR Pay is a specialist technology company that focuses on mobile payments using QR codes. It provides a smartphone application called QR Pal, which allows individuals and businesses to conduct payment transactions using QR code via the intermediate payment agent PayPal [147].

The literature has also confirmed that online authentication solutions can be secured using a QR code within two-factor authentication mechanisms. Tandon et al. [92] proposed the embedding of a cryptographic OTP into a QR code as a means of achieving secure authentication in online banking. Another use of QR codes in online banking systems was proposed by Adsul et al. [148]. From the scanned QR code, the scheme generates a string that is a combination of the IMEI number of the client's registered mobile phone and a random number. The string is automatically inputted into the login page in order to open the bank's homepage. The server decrypts the submitted string using the registered user's public key for verification. This method enhances the security level of banking by moving the authentication process from the web browser to the mobile phone, forcing the client to perform mobile banking. Furthermore, another QR code security enhancement mechanism has been proposed for use with ATM machines. In transferring money from one ATM card to another, a QR code confirmation pop-up message appears for scanning; this decodes recipient information for verification. Such an approach adds an extra security layer by employing mobile phones as a second authentication factor [148]. As attackers can easily collect financial credentials through web phishing, Kale et al. [149] suggested a secure single-sign-on (SSO) authentication scheme for untrusted computers via QR codes. The scheme is based on generating an encrypted QR code following the submission of the user ID. After decrypting the scanned QR code, the user's OpenID is requested by the mobile phone to generate a QR code, which will then be transferred to the untrusted computer for verification to finally gain access to the website.

Although most proposed authentication solutions are focused on embedding OTPs into QR codes on the server-side [92, 94, 98, 136], another approach proposed by Malik et al. [129] uses a timestamp as a verification criterion on both sides. During the scheme's enrolment phase, two personal assurance messages called PAM image and PAM text are selected by the user. Next, a digest based on the user ID, along with a secret key, are created and sent to the client. The verification phase displays an encrypted QR code that is based on the PAM image, PAM text, and a random number. The server also sends its timestamp, hash, and QR code contents to the client for a validity check. User validity is also checked. The client generates an OTP that is based on his registered identity and his timestamp, which is hashed along with the random number received. The user then sends the timestamp, along with its hash, to the server for verification. This scheme provides safe access across the web, with no replay attacks or MITM attacks due to timestamp and OTP cryptography, respectively. Moreover, the secret key is secured from intruders due to the use of a one-way hash function that generated the digest.

Other technologies can also be combined with QR codes and OTPs in authentication schemes. Liu and Zhu [103] proposed the utilization of some of the most important features of mobile phones in online authentication, namely the camera and Bluetooth, in combination with the QR code and OTP technology. Their scheme is based upon submitting the Bluetooth device address to the server during the registration phase as a means of connecting the current computer with the mobile phone. The QR code is used in the registration and authentication phases, while Bluetooth is used to submit the server-generated OTP and timestamp hashing result to the mobile phone during the login and authentication phases, respectively. The proposed protocol draws on a self-verified timestamp and QR code technology to prevent

MITM and replay attacks. OTP is used to increase security, and Bluetooth is used to reduce consumption. Moreover, a large amount of computation is not required due to the use of the one-way hash function and QR code cryptography. However, compared to a wireless connection, Bluetooth is weak and vulnerable to pairing failure. Besides online authentication, QR codes can be used to conduct financial transactions by simulating the mobile phone as a digital wallet. Ugwe and Mesigo [128] developed a model that uses QR codes as a communication link between parties engaged in financial transactions. The model is based on both parties launching their mobile wallet application. The receiver (merchant) selects the required payment amount and currency. Then, a QR code containing the account details and the necessary transaction information is generated. Once the payer (client) scans the QR code, the payment and transaction information are displayed, along with options to select which bank account or credit card to use for payment. After being authenticated via the mobile application, the payer is transferred to the selected financial institution's platform (such as PayPal) for the second authentication. Although the scheme is useful, both participants must be face to face in order for the transfer to take place.

2.4.2.3 QR Code Security Issues

QR codes have gained increasing popularity through their adoption in different fields of application, ranging from marketing [150] to secure mobile payments [135-138]. As a result, this technology has caught the attention of the hacking community, causing attackers to try to manipulate and modify its contents in order to advance their unethical pursuits.

QR code manipulation can be divided into two attack models. The first attack model is called "Both Colours," which has the ability to invert any module from black to white and vice versa. It generates a QR code sticker with the manipulated QR code to be positioned over the original code within the advertisement [127, 150]. This type of attack is trivial and represents an attack against a real-life advertisement, which is beyond the scope of this thesis. The second attack model is called "Single Colour," which is the basis for all QR code attacks. It is restricted to the modification of a single module from white to black but not vice versa, using a pen [127, 150].

2.4.2.4 Attacking Different Parts

As QR codes consist of different mandatory sections that contains a large amount of meta information, such as masks, version, and source code. Several different QR code regions can be targeted and manipulated by the attacker, either individually or in combination, in order to change the encoded information [127, 150].

- ***The Masks***

According to a particular rule, masks are used to generate evenly distributed modules in a QR code. This step helps modify QR code to make it easy for the QR code reader to scan and decode it [150]. The QR code specification standards [151] have defined eight mask patterns to be applied to the generated QR code, with the result of each being rated. The mask with the best distribution result based on the rating is chosen [127, 150].

Each mask pattern uses a specific formula to decide whether or not to change the colour of the bit. Table 2.4 lists the mask patterns formulas.

Table 2.4: Mask Pattern Formulas

Mask number	Formula (condition)
0	$(row + column)\ mod\ 2 == 0$
1	$(row)\ mod\ 2 == 0$
2	$(column)\ mod\ 3 == 0$
3	$(row + column)\ mod\ 3 == 0$
4	$\left(floor\ \left(\frac{row}{2}\right)\right) + \left(floor\left(\frac{column}{3}\right)\right)\ mod\ 2 == 0$
5	$(row * column)\ mod\ 2 + (row * column)\ mod\ 3 == 0$
6	$\left((row * column) mod2 + (row * column) mod3\right)\ mod\ 2 == 0$
7	$\left((row * column) mod3 + (row * column) mod2\right)\ mod\ 2 == 0$

If the conditions in Table 2.4 are true for a given module coordinate, the bit colour is changed [127, 150]. As mask patterns specifically apply to data modules and error correction modules, targeting or attacking the mask can affect these modules [127, 150].

- ***The Character Encoding (mode)***

The encoded data within the QR code starts with a four-bit mode indicator, which identifies the encoding mode of the information contained within the QR code [127, 150]. Table 2.5 lists each mode along with its indicator. If the mode indicator is changed or attacked, a whole new meaning is given to the encoded data [127, 150].

Table 2.5: Character Encoding Mode

Mode Name	Mode Indictor
Numeric Mode	0001
Alphanumeric Mode	0010
Byte Mode	0100
Kanji Mode	1000
ECI Mode	0111

- ***Character Count Indicator***

After the mode indicator is the character count indicator, which gives the number of characters being encoded within the QR code, represented as a string of bits. The size of the character count indicator depends on the encoding mode being used and the QR code version [127, 150]. Table 2.6 lists the sizes of the character count indicator for each encoding mode and version.

Table 2.6: Character Count Indicator size for each Encoding Mode and Version

Version	Numeric	Alphanumeric	8-bit	Kanji
1-9	10	9	8	8
10-26	12	11	16	10
27-40	14	13	16	12

Two main approaches can be used to attack the character count indicator, producing buffer underflow or buffer overflow [127, 150].

Buffer Underflow: Changing the number representing the character count indicator to a smaller number causes buffer underflow. Therefore, the QR code reader will only decode the number of characters specified by the attacker, causing it to neglect the rest of the characters and treat them as a new segment or as fillers. This kind of attack can be useful if the encoded data was a URL link with suffixes. Nevertheless, QR codes can overcome this type of attack and still be decodable. If only a small change is made, this will result in a small change in the error correction values [127, 150].

Buffer Overflow: Changing the number representing the character count indicator to a higher number causes buffer overflow. Therefore, the QR code reader tries to decode parts of the fillers as data. This type of attack is difficult to perform in a real-life scenario, due to the abilities and great powers of the employed Reed-Solomon Code [127, 150].

- ***Mixing Modes***

It is possible to mix different modes into a single QR code through concatenation, which is useful when encoding different types of data. This is done by concatenating several data segments, along with their mode indicators and character count indicators [127, 150]. Table 2.7 shows how different data segments can be concatenated within a QR code.

Table 2.7: Concatenating different data segments within a QR code

Segment 1			Segment 2				Segment n		
Mode Indicator 1	Character Count Indicator 1	Data	Mode Indicator 2	Character Count Indicator 2	Data	…	Mode Indicator n	Character Count Indicator n	Data

This feature aids attacks using four different approaches:

1. **Changing mode of segments**: This works the same way as attacking the encoding mode of a QR code. However, in the mixed mode, the change is limited to one segment only, while the other data are left untouched [127, 150].

2. **Inserting new segments**: A segment can be split into two new segments, one with the same header (the same mode indicator and character count indicator), and one with a new header [127, 150].

3. **Deleting existing segments**: When a segment is deleted, the space used by the mode indicator and the character count indicator is overwritten with additional data. Moreover, the character count indicator of the previous segment can be changed in order to append data to the deleted segment [127, 150].

4. **Structured append**: This mode allows concatenation of up to 16 QR codes. Decoding these QR codes is done independently of their encoding order. Attackers

exploit this feature through adding a segment pointing to another QR code, which is beneath the original segment. This attack is less practical due to the complexity of the structured append mode and the preparation overhead [127, 150].

- ***Data and Error Correction***

As illustrated in Figure 1, data and error correction codewords represent the largest part of the QR code. These two parts are related in a way that only changes when the actual data is directly reflected in the data and error correction portions [127, 150]. The Reed-Solomon encoding used to convert the data into data bits [142] can detect changes in either the data or error correction parts and decodes the original message even if some modifications or errors have been made. Hence, it can be said that the error correction feature of the Reed-Solomon code increases the integrity of the QR code as much as possible [127, 150].

2.4.2.5 QR Codes as Attack Vectors

Whether the QR code reader is human or an automated process, manipulated QR codes can be the basis for a number of attacks [127, 150].

2.4.2.5.1 Attacking Automated Processes

The consequences of scanning malicious QR codes can go beyond the victims who scan these codes; such attacks can affect the backend system that is trying to serve as an intermediate between the requests generated upon QR code scanning and the client [142]. For example, SQL injection is capable of manipulating the backend database by appending a semicolon followed by an SQL query to the encoded information, which can result in a denial-of-service attack in the case that the SQL query was deleting a table [127, 142, 150]. Another attack based on automated processes is command injection, which can cause severe damage to the operating system of the automated system. This attack uses the encoded information as a command line parameter without being sanitized [127, 142, 150]. In other words, a user who scans a random QR code created by an attacker will generate the attack on behalf of the attacker [127, 142, 150]. Furthermore, fraud can be committed via automated system changes [127, 150].

2.4.2.5.2 Attacking Human Interaction

As the encoded information within the QR code is completely obscure, humans cannot decide whether or not the code is malicious, even when decoded by the reader software. For example, a QR code might contain a URL of a phishing website that is similar to the original URL. This form of attack is called “Phishing and Pharming,” and is serious, especially if some form of credentials are required [127, 150]. In case of advertisements, QR codes can be manipulated by attackers to redirect users to cloned websites, in order to commit fraud [127, 150]. Other attacks are based on command injection or traditional buffer overflow as a means to attack the reader software on computers or smartphones. The attacker might take over control of the entire smartphone, including its contacts, e-mail or SMS [127, 150]. Based on these attacks, social engineering attacks are also possible, such as spear phishing or publishing manipulated QR codes in public places [127, 150].

2.5 Encryption Technology Layer

In e-commerce, encryption technologies are an essential means for protecting confidential information leakage [20]. Thus providing four of the six key dimensions of e-commerce security, namely integrity, non-repudiation, authentication and confidentiality [24]. Over the years, a large number of cryptographic algorithms have been developed for use in many different functions and security protocols [152] to ensure secure communications [41]. Most current communication systems rely on security protocols and the properties of secrecy and authentication [41].

Since cryptography is considered the “science of keeping secrets secret” [153], and secrecy is one of the mandatory and fundamental security properties in any secure protocol [41], cryptography is considered one of the fundamental aspects in the field of computer security [154].

According to the National Institute of Standards and Technology (NIST), cryptography is defined as “a branch of mathematics that is based on the transformation of data and can be used to provide several security services: confidentiality, data integrity authentication, authorization, and non-repudiation" [155]. In terms of cryptology science, cryptography is a part of a wider science called "cryptology" [156, 157], which is the "science of information security and privacy" [156]. It is divided into two main branches: cryptography and cryptanalysis [156]. Cryptography is the design part of the science [156], while cryptanalysis involves analysis and security investigation [156].

Cryptography is based on an algorithm (cipher) and a key, which are used to encrypt and decrypt the data [155]. In general, the algorithm is a mathematical function that transforms the data from plaintext to ciphertext, and from ciphertext back to plaintext, with the help of the key parameter [155].

The types of cryptographic algorithms fall into three broad categories: symmetric cryptography, asymmetric cryptography, and cryptographic hash function [153-155, 157, 158].

2.5.1Symmetric Cryptography

In symmetric cryptography, also referred to as conventional encryption [159], secret-key schemes, or single-key schemes [154, 155, 157], the same secret key is used for both encryption and decryption [153-155, 157, 160]. Examples of such cryptography include the Advanced Encryption Standard (AES) [152] and Triple Data Encryption Standard (3DES) [152].

Symmetric key algorithms are usually very secure [157] and fast [157, 160]. However, several shortcomings are associated with symmetric key schemes, such as:

- **Secret-key distribution**: Prior to the sender and receiver communicating with such a cipher, the secret key must be exchanged among participants in a secure manner [153, 157, 160]. Since the communication link is not secure, there is always the risk for the secret key to be intercepted by an adversary, causing messages to be decrypted and impersonation of users [157, 160]. Therefore, secret-key distribution is considered the biggest problem in symmetric cryptography [158].

- **Number of keys**: In a network corporation between n users, if each pair of users needs a separate key-pair, then each user will have to save $(n-1)$ keys, and an $n.\frac{n-1}{2}$ key pair must be generated and transported over a secure channel [157].

- **Repudiation**: Either one of the participants can deny involvement in an e-commerce application. Therefore, symmetric cryptography cannot be used in applications that need to satisfy the non-repudiation requirement [157]. In these cases, asymmetric cryptography should be employed through a digital signature [158].

Symmetric cryptography is divided into stream ciphers and block ciphers [157]. A stream cipher is a method that encrypts plaintext into ciphertext by applying a cryptographic key and an algorithm to each bit of the plaintext, individually [157]. The method works by generating a keystream, which is a pseudorandom sequence of data. Next, each bit of the keystream is XORed with the corresponding bit of the plaintext to generate the ciphertext. To decrypt, the same method is applied, where the keystream is XORed bit-by-bit with the corresponding bit of the ciphertext [161].
A stream cipher is very fast, compact and efficient [157]. It is relevant for applications with very low hardware complexity, such as cell-phones and other embedded devices, where few processor cycles and small chip areas are required for encryption [157]. However, stream ciphers are not used in modern cryptography as often as block ciphers [157].

A block cipher can be either symmetric or asymmetric [162]. Therefore, it is considered the most important and dominant element in many cryptographic protocols [162]. A block cipher can be defined as a mapping function [162] that operates on blocks of data [163]. It encrypts a block of n-bits of plaintext into a block of n-bits of ciphertext by applying a cryptographic secret key and an algorithm to the block at the same time [157]. Such blocks are usually 64-bits (8-bytes), e.g., the Data Encryption Standard (DES) or the Triple DES algorithms (3DES), or 128-bits (64-bytes), e.g., the Advanced Encryption Standard algorithm (AES) [157, 162, 163].

In terms of building blocks, a block cipher represents the fundamental building block for other major cryptographic tools. A block cipher can be turned into a stream cipher or a hash function, which plays a major role in cryptographic protocols [162]. In addition, the method satisfies data confidentiality and authenticity [162]. In terms of speed and compactness, stream ciphers can be used to perform a fast cipher to encrypt a large amount of data due to having a smaller size [157] and faster speed [157, 164] than block ciphers, and the method is more efficient for hardware and software because it requires few processor instructions and few gates [157], respectively. However, modern block ciphers are also very efficient when it comes to hardware and software [157].

2.5.1.1 Data Encryption Standard (DES)

The Data Encryption Standard (DES) is an old symmetric block cipher algorithm designed by IBM in 1972 [165, 166]. DES is an iterated block cipher [153, 160] that operates on blocks of plaintext of 64-bits with 16 rounds of encryption, using a shared key of 56-bits [165, 166].

The DES algorithm is based on the Feistel structure, in which each block is split into two to be fed into the Feistel network [157]. The structure feeds the right half of the input into function f, where its output is XORed with the left half. Thus, only the left half of the input is encrypted, while the right half moves on to the next round unchanged after swapping its position with the right half. In each round, a different key is derived from the main 56-bit shared key using a key schedule [157]. One of the advantages of DES is that the decryption and encryption functions are the same; the decryption function reverses the encryption in a round-by-round technique [157]. On the other hand, the DES key space is too small to be used to encrypt confidential data and is thus vulnerable to brute-force attacks [157]. Furthermore, DES is very slow when implemented in software. However, variants of DES, such as 3DES, are much more secure [157].

2.5.1.2 Triple Data Encryption Standard (3DES)

In 1998, DES was replaced by Triple-DES (3DES); this was done to overcome DES's deficiencies without changing its original structure [165]. 3DES achieves a higher level of security in data encryption because it performs three iterations of DES on each block, using three different unrelated keys [165]. As with its predecessor, 3DES is based on the Feistel structure, and it operates on blocks of plaintext of 64 bits with 48 rounds of encryption, using a key of size of 168 bits that is permuted into 16 subkeys [167]. Although 3DES resists brute-force attacks, it still has several problems with regard to software implementation, including lack of efficiency and speed, with 3DES being three times slower than DES [157]. Another shortcoming is its small block size, which is considered a drawback in some applications [157]. Moreover, 3DES is exposed to differential and related key attacks [165].

2.5.1.3 Advanced Encryption Standard (AES)

Due to the disadvantages of DES and 3DES, the National Institute of Standards and Technology (NIST) looked to develop a new block cipher to replace DES. Therefore, a formal call for a new Advanced Encryption Standard (AES) was announced by NIST in 1997 [157]. In 2001, Rijndael was chosen from among five finalist algorithms (Mars, RC6, Rijndael, Serpent, and Twofish) as the new AES [157].

AES operates on blocks of 128 bits, with 10, 12, or 14 rounds, using key sizes of 128, 192, or 256 bits, respectively [167]. The reason for having small number of rounds is that AES encrypts an entire block per iteration. The AES round functions deals with all 128 bits of the data path (also called algorithm state or state matrix), arranged in a four-by-four byte matrix [153, 157].

Each round of the algorithm (except for the last) consists of four steps: Byte Substitution, ShiftRows, MixCoulmn and Key Addition. For each individual AES round, Byte Substitution starts by feeding the 16-byte data input into the S-box for transformation [153, 157]. The S-box uses a look-up table with special mathematical properties [157]. The generated 16-byte output is then transformed in the ShiftRows step through a periodic left shift of the rows within the state matrix (except for the first row) to increase the diffusion property [157]. Another transformation is performed within the MixCoulmn step by multiplying each column of the state matrix by a constant matrix, where each input byte affects another four output bytes. Thus, MixCoulmn is considered the primary diffusion element within the AES algorithm [153, 157]. Finally, the Key Addition step is a bitwise XOR operation between the intermediate 16-byte state matrix and a 16-byte subkey to generate the next subkey. Note that the first subkey used in the initial round is the original AES key, and the following AES subkeys are derived via a key schedule [157].

Since AES is not based on the Feistel network, all steps are invertible as a means of performing decryption [157]. AES also has the ability to provide strong diffusion, confusion, and long-term security against brute-force attacks [157].

In a comparative analysis of symmetric algorithms done by Ebrahim et al. [165], AES was observed to be superior in terms of security memory usage, flexibility, and encryption performance. It has also been shown to be efficient with regard to software and hardware implementation [157]. Therefore, AES has become the compulsory encryption algorithm in many commercial systems, including the internet security standard IPsec, TLS, the Wi-Fi encryption Standard IEEE 802,11i, the internet phone service Skype, and many other security products [157]. Furthermore, by 2003, the US National Security Agency (NSA) was using AES to encrypt classified documents up to top secret level [157], and it was also being used for other US government applications [157]. However, based on the number of rounds and the key size, a large amount of memory is required for storage. Therefore, it is undesirable to implement AES on devices with limited memory resources [157].

2.5.2Asymmetric Cryptography

In asymmetric cryptography, also referred to as public-key cryptography [154, 155, 157], two mathematically related keys—the public key and private key—are used for the encryption and decryption processes [153-155, 160]. The public key can be widely published and used publically by everyone to encrypt data [155, 160], while the private key must remain secret, known only by its owner, to be used for decryption [155, 160]. Although the two keys are related, the private key cannot be extracted from the public key [155].

Based on symmetric cryptography, the idea behind asymmetric cryptography is the importance of maintaining the confidentiality and secrecy of the decryption key, and not the encryption key [157]. Therefore, in their landmark paper, Diffie, Hellman, and Merkle proposed the revolutionary "public-key cryptography" [168], in which the public key is published to be used by everyone, and while its matching private key is secured for decryption [157].

In terms of security mechanisms, public-key cryptography has been designed in a way that it provides secure cryptographic schemes that satisfy data integrity, non-repudiation, and authentication [155]. In the case of symmetric cryptography, a message can be encrypted without a secret channel for key establishment, but much effort is required to manage those keys. This protocol can be modified using public-key cryptography [157]. In this modification the symmetric key is encrypted with the public-key, and once the receiving party decrypts the symmetric key with the private key, both parties can exchange messages using symmetric cryptography [157]. This methodology combines the best features of both symmetric and asymmetric cryptography [157].

Besides using public key schemes to encrypt data, asymmetric cryptography provides the following security mechanisms:

- **Key establishment**: There exist a number of public-key protocols to establish secret keys in a secure manner over an insecure network [157]. Examples of such protocols include the Diffie-Hellman key exchange or RSA key-transport protocols [157].
- **Non-repudiation**: The repudiation problem can be overcome through a digital signature [155, 157, 160] that is recognized by the asymmetric cryptography algorithms [160]. The methodology is based on encryption of the data with the private key, which is later verified by the corresponding public key through decryption [160].
- **Identification**: Identification of entities can be achieved through digital signatures within challenge-response protocols [157], such as those used in online banking.

Although symmetric cryptography can identify entities and perform encryption, much effort is required for key establishment [160]. Moreover, it poorly satisfies the non-repudiation requirement [157, 160]. Asymmetric cryptography, on the other hand, seems to satisfy the requirements of modern security protocols, such as data integrity, non-repudiation, and authentication [155], and provides key establishment functionality [155, 157]. However, its encryption mechanism is very slow [157]. Nevertheless, in order to take advantage of the best features of both methodologies, it is recommended to merge both symmetric and asymmetric algorithms, as is done in the SSL/TLS protocol [157].

2.5.2.1 RSA

The RSA algorithm (Rivest-Shamir-Adleman) is the most popular and widely used asymmetric cryptosystem, and is based on public key cryptography [157]. The algorithm uses a pair of keys—a public and a private key—to perform the encryption and decryption processes, respectively [160, 169]. The RSA algorithm consists of three main steps: key generation, encryption, and decryption [170]. The algorithm is computationally intensive, since it is based on generating and publishing a public key based on two large prime numbers (p, q) that are used to compute the public and private keys. The prime numbers must be kept secret as a means to prevent private key disclosure.

In practice, RSA is slow compared to symmetric cryptography due to the complexity of the operations performed during execution [160]. Therefore, it is mainly used to securely encrypt and transport session keys in symmetric ciphers such as AES or 3DES due to its time complexity [157, 160]. It is also used in digital signatures through encrypting messages with the private key, to be decrypted in such a case with the public key [157, 160]. Furthermore, it is used in the key exchange algorithm within the TLS/SSL protocol [167, 171, 172].

In terms of security, RSA has an undesirable property termed "malleability," which allows an attacker to transform a ciphertext into another ciphertext, leading to a known plaintext transformation. Malleability only allows manipulation of the plaintext without decryption, which can cause harm, especially in the financial world. However, padding provides the solution to malleability; it involves embedding a random structure into the plaintext prior to encryption [157]. Other shortcomings of RSA involve the operational complexity of the execution, which causes the RSA cryptosystem to be much slower than DES, 3DES, and AES [160, 173]. Moreover, some chip-card microprocessors cannot employ RSA cryptosystems in their applications due to its inability to perform computations with keys longer than 1024 bits [160].

Furthermore, RSA is faced with two problems that represent the bases upon which the algorithm's security and cryptographic power rely [157, 160]. The two problems are based on their mathematical structure; they are the RSA problem and the integer factorization problem [160]. The RSA problem revolves around recovering the plaintext from the ciphertext without actually having the private key, but only knowing the public key(e, n). One possible approach to solving the RSA problem is to factor the modulus n, which can retrieve the prime numbers (p, q). The factoring approach makes it easy for the attacker to compute the private key (d, n) [160, 174]. Therefore, it is stated that the "RSA and integer factorization problems are computationally equivalent" [160, 174]. Thus, prime numbers (p, q) must be chosen in such a way that makes it impossible for any algorithm to break the

RSA. In addition, secure keys must be generated that cannot be broken for at least two decades, in the case that reasonable long-term security is required. In other words, prime numbers must be sufficiently large enough to generate an integer n that has 2048 bits in its binary representation. The need for numbers with thousands of bits is due to the fact that numbers with hundreds of bits are now factorable [160].

2.6 Security Authentication Layer

Secure authentication technologies are the first line of defence against security threats [24]. It represents the primary means for verifying the identity of the participants within an e-commerce system, along with ensuring the authenticity of the exchanged messages. Secure authentication technology include digital digest, digital envelope, digital signature, digital certificate, certification authority, digital timestamping, etc.
Among the previous technologies, the most suitable solutions serving the purpose of the proposed MUAS are digital digest, digital certificate and certification authority.

2.6.1Digital Digest (Hashing)

Hashing is an essential cryptographic primitive that is widely adopted in information security protocols [157, 175, 176] to provide message integrity [177]. It is a computationally feasible function that computes a fixed-length message digest from an arbitrary-length message [157, 175]. Although the use of hash functions within modern cryptography is manifold, hashing is best known for its use in digital signatures [157], which are signed cipher texts that are sent over the internet as a means to validate the integrity and authenticity of a message, as well as to ensure non-repudiation [176].

Currently, the most popular cryptographic hash functions are SHA-1 (Secure Hash Algorithm 1) and SHA-2, which replaced their predecessor MD5 (Message Digest) [160]. MD5 is a cryptographic hash function that produces a 128-bit message digest. It was widely used in internet security protocols for storing password hashes and computing files checksums. However, MD5 also has potential weakness, in that collision attacks could occur in some cases. Wang et al. [178] published an effective method against the MD5 algorithm, in which two different messages evaluate the same MD5 hash value. Thus, a secure hash algorithm (SHA) was published by the US NIST in 1993 [157]. SHA-0 was the first member of the SHA family; it was later modified to become SHA-1, in which the cryptographic security was improved by modifying the schedule of the compression function [157]. Both SHA-0 and SHA-1 produce a 160-bit message digest. Later, a partial attack was conducted on MD5 [179], leading SHA-1 to become widely adopted in a large number of products and standards [157]. However, a number of collision search attacks on the SHA-1 algorithm were carried out by Wang et al. [178] and Stevens [180], providing a useful perspective on the design criteria for a more secure hash algorithm.

In 2001, the US NIST Federal Information Processing Standards published SHA-2, the second member of the SHA family [157, 160]. SHA-2 consists of four hash algorithms, SHA-224, SHA-256, SHA-384, and SHA-512, with output message digests of 224, 256, 384, and 512 bits, respectively [157, 160].

The SHA-1 and SHA-2 family algorithms are based on a very similar design principle. Compared to MD5, both algorithms are less prone to brute-force attack and differential cryptanalysis [160]. In terms of the output digest, SHA-1 produces a 160-bit message digest, while SHA-2 produces a larger digest, which in turn makes the algorithm more secure and harder to break [160].

According to the Secure Hash Standard (SHS) from the National Institute of Standards and Technology (NIST), the seven approved algorithms are SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256. When inputting a message that is shorter than 2^{64} bits (for SHA-1, SHA-224, and SHA-256) or 2^{128} bits (for SHA-384, SHA-512, SHA-512/224, and SHA-512/256) into a hash algorithm, the output is called a message digest; it ranges in length from 160 to 512 bits, depending on the algorithm in use [181]. Table 2.8 lists the basic properties of the seven hash algorithm [181].

Each algorithm consists of two phases: pre-processing and hash computation [157, 160, 181]. The pre-processing phase involves padding the message into a multiple of 512 or 1024 bits [157, 160, 181], dividing the padded message into message blocks [157, 160, 181], and setting the initial hash value to be used in the hash computation [157, 160, 181]. The hash computation phase uses the padded message to generate a message schedule, which is used along with functions, constants, and word operations to generate a series of hash values. The final value is the message digest [157, 181].

Table 2.8: Secure Hash Algorithm Properties

Algorithm	Message size (bits)	Block size (bits)	Word size (bits)	Message Digest size (bits)
SHA-1	$<2^{64}$	512	32	160
SHA-224	$<2^{64}$	512	32	224
SHA-256	$<2^{64}$	512	32	256
SHA-384	$<2^{128}$	1024	64	384
SHA-512	$<2^{128}$	1024	64	512
SHA-512/224	$<2^{128}$	1024	64	224
SHA-512/256	$<2^{128}$	1024	64	256

2.6.2Hash Algorithm Security

The security strength of hash functions is defined by three main properties that must be met in order for the hash function to be considered secure [157].

1. **Pre-image Resistance (One-Wayness)**: This property states that it is computationally infeasible to derive the input message from the hash value. That is, given a hash value $hash\ value$, it is computationally infeasible to derive the hash function's input message x [157, 160, 175, 182], such that $hash(x) = hash\ value$ [175]. Therefore, this is called a "one-way" hash function [157, 182]. The amount of work needed to obtain a pre-image for a hash function is measured by the length of the $hash\ value$ produced by the hash function, i.e., for an L-bits hash function, the expected pre-image resistance is L bits [175].

2. **Second Pre-image Resistance (Weak Collision Resistance)**: This property states that it is computationally infeasible to find a second input message that is different than the given first input message, with both evaluating to the same hash value [157, 175, 182].

That is, given an input message $x1$, it is computationally infeasible to find a second input message $x2$, where $x1 \neq x2$, such that $hash(x1) = hash(x2)$ [157, 160, 175, 182]. The amount of work needed to find a second pre-image for a hash function is measured by the length of the $hash\,value$ produced by the hash function, i.e., for an L-bits hash function, the expected second pre-image resistance is L bits [175].

3. **Collision Resistance (Strong Collision Resistance)**: This property states that it is computationally infeasible to find two different input messages that evaluate to the same hash value [157, 175, 182]. That is, it is computationally infeasible to find two different input messages $x1$ and $x2$, such that $hash(x1) = hash(x2)$ [157, 160, 175, 182]. The amount of work needed to find a collision for a hash function is measured by half the length of the $hash\ value$ produced by the hash function, i.e., for an L-bits hash function, the expected second pre-image resistance is $L/2$ bits [175].

Table 2.9 summarizes the strength of the security properties for each one of the approved hash algorithms [175].

Depending on the properties required of the hash function, the security strength is determined by either one of the previous three main properties. If more than one property is required from a hash function, the security strength of the hash function is defined by the weakest property [175]. For instance, a digital signature requires both collision resistance and second pre-image resistance from the hash function. Therefore, since the collision resistance strength $(\frac{L}{2})$ is less than the second pre-image resistance (L), the security strength of the hash function used for a digital signature is defined as its collision resistance strength [175].

Table 2.9: Strengths of the Security Properties of the Hash Algorithms

	SHA-1	**SHA-224**	**SHA-256**	**SHA-384**	**SHA-512**	**SHA-512/224**	**SHA-512/256**
Pre-image Resistance in bits	160	224	256	384	512	224	256
Second Pre-image							

Resistance in bits	105-160	201-224	201-256	384	394-512	224	256
Collision Resistance in bits	<80	112	128	192	256	112	128

2.6.3Public Key Infrastructure (PKI)

Among the advantages of asymmetric cryptography is to enable digital signatures, where data is electronically signed before being submitted [160]. Digital signatures are based on sharing the public key of the sender to the other party, either through an e-mail, a web site or any form of electronical exchange [160]. It ensure authenticity of the message and non-repudiation of the sender through encrypting the message with the sender's private key to be decrypted by the recipient with the public key of the sender [24]. However, an intruder can impersonate identities and replaces the transmitted public key with his own key, causing an incorrect authorization [160]. Hence, a trusted third party called certificate authority (CA) confirms the authenticity of the public key through signing it with its private key to authenticate the identity of both parties while exchanging data, thus achieving a secure and reliable system [160, 183]. Certificate authority is a trusted government agency or financial institution who issues digital certificates, which is a digital document containing the name of the subject, company and public key. In addition to certificate identifying information [24, 183].

Public key infrastructure system refers to the digital certificate issued by a CA. The process begins by the user generating a public/private key pair, and sending a request to the CA along with the public key. Following the CA verifying the submitted information, a certificate containing the received public key and other related information is issued. Eventually, the certificate is signed with the CA's private key, thus generating a message digest called signed certificate [24, 160, 183]. Whenever the certificate is published, any party can verify the real identity of the user by authenticating the certificate against the reliable CA's [183].

With respect to expanding and diversifying mobile services, it is essential to provide trustworthy remote access to security-sensitive systems such as internet banking and e-commerce. Therefore, a trend towards sharing certificates securely between devices has evolved. A novel certificate sharing system (CSS) was proposed by Kim et al. [184] to share certificates securely without modifying the existing standard format between a PC and a smart phone. The CSS is a special conversion module for storing the same X.509 certificate on both a PC and a smart phone. The CSS module converts the certificate on the PC to binary code to be saved on the CSS database server (DB server). The user begins the process by authenticating and issuing an X.509 certificate from the certification authority (CA) via the web server. Four parameters are required for authentication: user ID, password, certificate password, and security answer. Upon issuance, the certificate is transferred to the PC's CSS module and saved as a .CERT file. The device's unique information is then incorporated into this active certificate to prevent it from being copied.

To upload the certificate from the PC to the web server, the authenticated user submits the certificate's password for authentication within the CSS module. Once authenticated, the CSS module performs the following operations: ***(1)*** removes the unique hardware information from the .CERT file and encrypts the X.509 certificate via private key, ***(2)*** converts the .CERT file into a personal information change (.PFX) with a 192-bit secret key and temporarily saves the .PFX file in the root folder of the PC, and ***(3)*** converts the .PFX file into binary form using a 128-bit secret key, which will be uploaded into the webserver. After the binary code is converted, it is saved in the database server, and the .PFX file is deleted from the root folder. When the binary codes are successfully saved to the database server, an OTP is generated by the OTP server and sent to the user's smart phone.

To download the certificate from the DB server into the smart phone, the user authenticates his/her account via user ID and password, then submits the OTP and the certificate password as a request for downloading the binary form of the X.509 certificate. Once the binary code is downloaded, the smart phone's CSS module performs the following operations: ***(1)*** converts the binary code into a .PFX file using the 128-bit secret key, which will be saved to the root folder, ***(2)*** converts the .PFX file into a .CERT file with the 192-bit secret key, ***(3)*** deletes the binary code and OTP from the database server for certificate security and to prevent reuse and, ***(4)*** incorporates the device ID into the converted file, and saves the .CERT file in a specific folder on the smart phone. Once the .CERT conversion is complete, the .PFX file is deleted from the smart phone.

This approach provides strong end-to-end data security for the certificate, as well as strong data security on physical devices. Moreover, it is operable on low-end smart phones with older operating systems, and it can be flexibly applied to enterprise environmental systems.

2.7 Security Protocol Layer

Secure communications within E-commerce are an indispensable requirement to provide security attributes such as authenticity, integrity, confidentiality, privacy, availability and non-repudiation [43]. Thus, secure cryptographic protocols have been designed to protect data among the different network layers [185].

Transport Layer Security (TLS) and Internet Protocol Security (IPSec) are known cryptographic network security protocols that prevent data transmitted between participants from being eavesdropped on or tampered with [186]. The security features of these protocols are based on the use of cryptography [187]. TLS and IPSec are not interchangeable, as each protocol operates on different levels within the TCP/IP protocol stack [186]. Thus, many differences exist between them [43].

IPSec is a collection of protocols that operates at the network layer of the internet protocol suites (TCP/IP), as illustrated in Figure 2.2, providing transparent security for upper layer protocols [186, 188]. The layered structure of the TCP/IP prevents the protocol from changing its type of protection to conform to different applications [186]. Thus, IPSec does not provide user-to-user nor application-to-application security; it provides only host-to-host security [43, 186, 188].

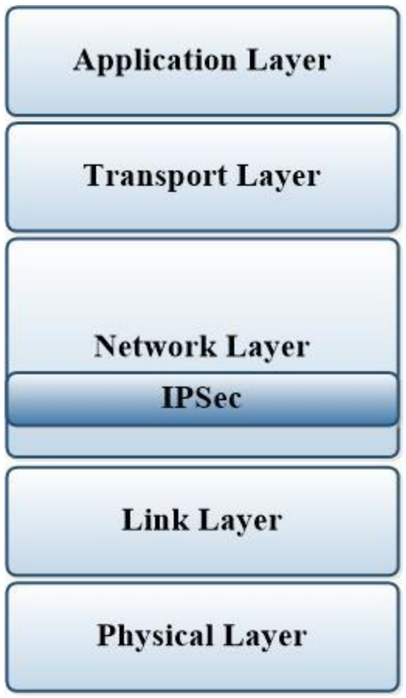


Figure 2.4: IPSec in the TCP/IP stack

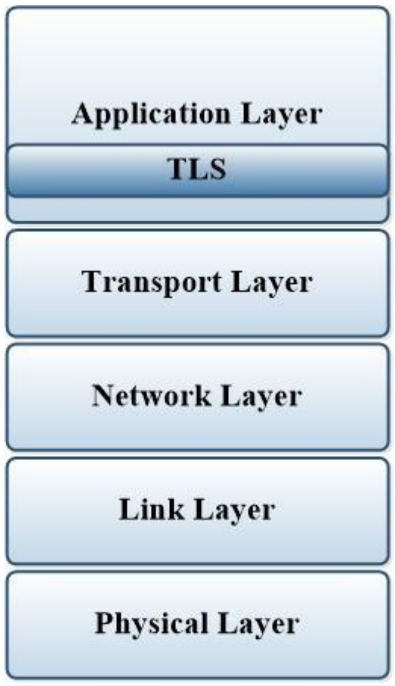


Figure 2.3: TLS in the TCP/IP stack

TLS, on the other hand, is an Internet Engineering Task Force (IETF) standard that deals with network security issues within the context of E-commerce [43]. It operates at the application layer of the internet protocol suites (TCP/IP), as illustrated in Figure 2.3. It provides application-to-application security [186, 189].
TLS is used to secure communications and protect sensitive data in a wide variety of online transactions, including financial, healthcare-related, and social [189]. It uses a set of cryptographic algorithms to ensure data authenticity, integrity, and confidentiality [189].

The proposed authentication mechanism, which connects a web browser (client) to a server in the context of E-commerce, uses a mobile phone application as the second factor to generate and send the response back to the server. Taking into consideration the security requirements of this approach, TLS is the most suitable protocol for use within the proposed MUAS. TLS is the security backbone of many client-server applications [172].

2.7.1TLS/SSL

TLS stands for “transport layer security.” It is a client/server protocol that exists in all major web browsers [187] to secure data transmitted between any two communicating applications within the application layer [189]. TLS is application independent [189, 190], which ensures data privacy and integrity [190, 191] between applications running on top of the transport layer [172]. The reason for placing TLS at higher levels is to help secure communication between the client's browser and the server's application. Thus, it is more appropriate for web-based applications [167, 192].

The TLS protocol is divided into two layers [191, 193]. The first layer is composed of the application protocol and three handshake sub-protocols: the Handshake protocol, the Change Cipher Spec protocol, and the Alert protocol. The second layer contains the TLS Record protocol. [43, 186, 191]. Figure 2.4 shows the different protocols constituting the TLS protocol [191].

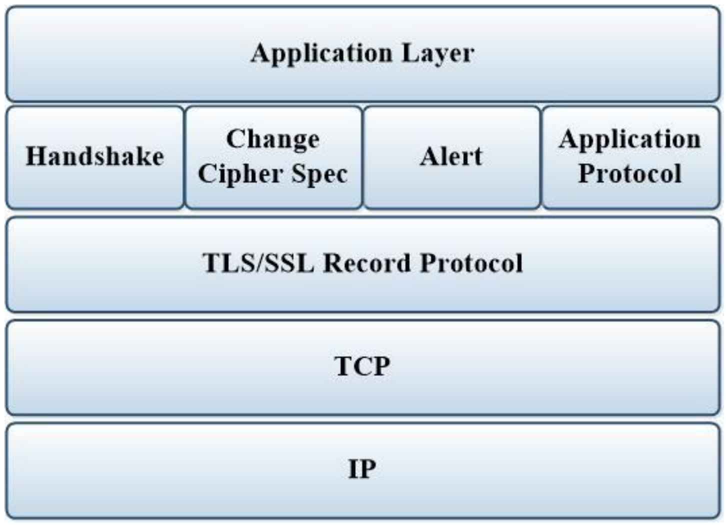


Figure 2.5: TLS Protocol stack

- The TLS Handshake sub-protocol is responsible for establishing and resuming secure sessions between clients and servers by negotiating session parameters [171, 191, 193]. Figure 2.6 shows the initial negotiation and establishment of secure session parameters.

- The TLS Change Cipher Spec sub-protocol is a single one-byte message, exchanged by between parties to indicate the transition to a new negotiation to update the Cipher Spec and keys [171, 191].

- The Alert sub-protocol is responsible for reporting error conditions to the other party [171, 193].

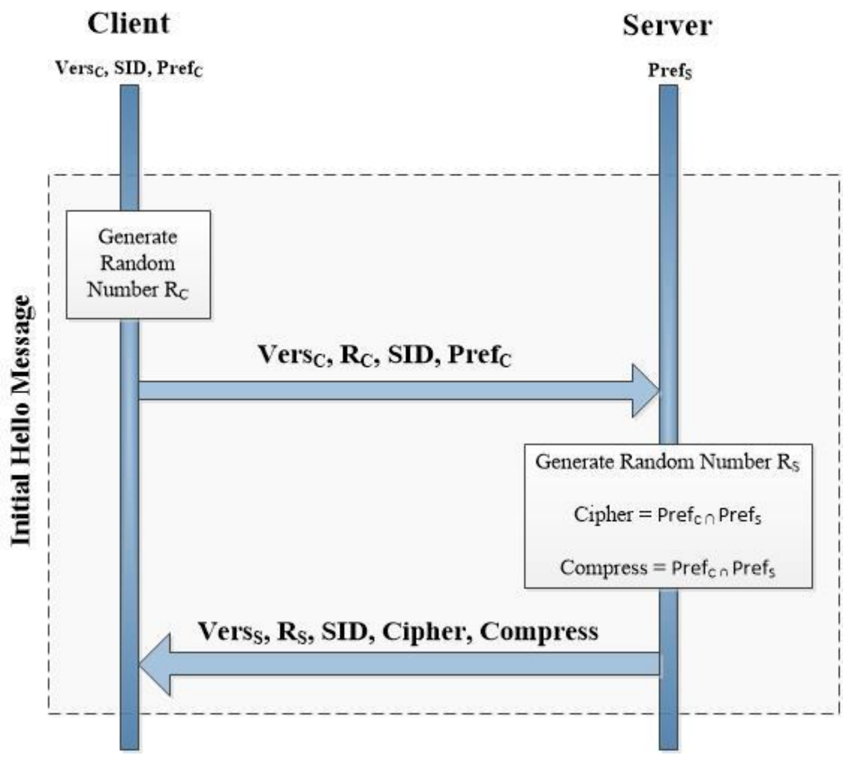
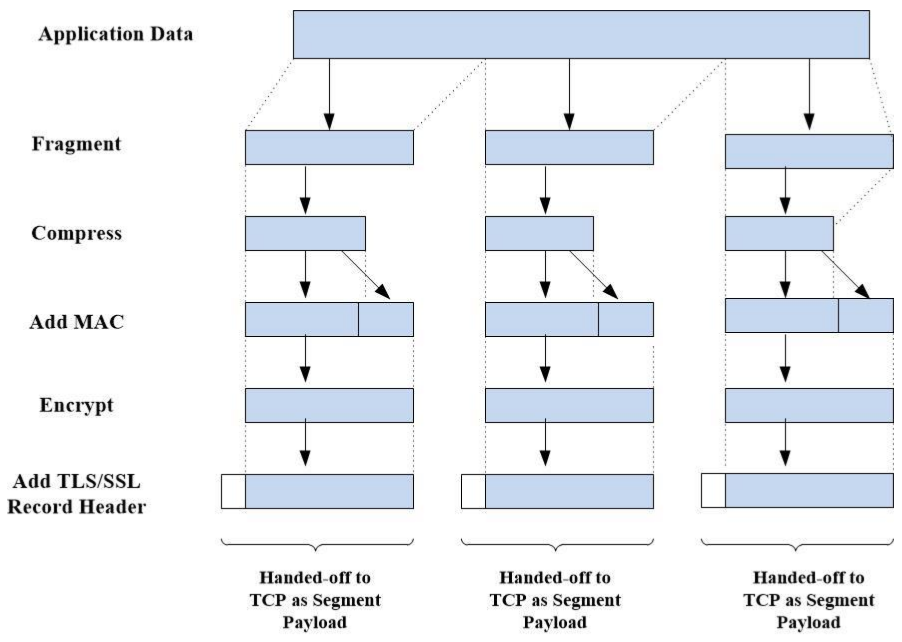

Client
$Vers_C$, SID, $Pref_C$
Server
$Pref_S$
Initial Hello Message
Generate Random Number R_C
$Vers_C$, R_C, SID, $Pref_C$
Generate Random Number R_S
Cipher = $Pref_C \cap Pref_S$
Compress = $Pref_C \cap Pref_S$
$Vers_S$, R_S, SID, Cipher, Compress

Figure 2.6: TLS Initial Negotiation

The TLS record protocol is responsible for ensuring the authenticity, integrity, and confidentiality of the exchanged application data through the use of a set of cryptographic algorithms [193]. The operations include fragmentation, compression, application of MAC, and finally, symmetric encryption to be passed on to the TCP layer after adding the TLS record header [43, 172]. Initially, the MAC and encryption key parameters are initialized to NULL. Therefore, the protocol does not use MAC or encryption during the initial handshake negotiation phase [172]. Figure 2.7 clarifies the TLS record protocol operations [43].



Application Data
Fragment
Compress
Add MAC
Encrypt
Add TLS/SSL Record Header
Handed-off to TCP as Segment Payload
Handed-off to TCP as Segment Payload
Handed-off to TCP as Segment Payload

Figure 2.7: TLS Record Protocol Operations

Most TLS studies have focused on the initial key exchange phase within the handshake protocol, while potential re-negotiation has not been investigated in depth. However, Gelashvili [172] did conduct research on the key re-negotiation phase, pointing out some of the functions that makes TLS one of the most popular and secure cryptographic algorithms. Based on Gelashvili's study, the TLS handshake protocol can be classified into three successive phases: the setup phase, the re-negotiation phase, and the finishing phase. The TLS setup phase is responsible for setting-up the session's security parameters [186]. The session parameters consist of the TLS version $(Vers)$, a session ID (SID), random values (R_C, R_S) and cipher specification $(Pref)$ [172, 186, 191]. Each cipher defines a key exchange algorithm, an encryption algorithm, a MAC algorithm, and a pseudo-random function for key generation [172, 191]. The selected key exchange algorithm defines the precise structure of the entire negotiation phase regarding whether the server will start with a server certificate authentication, or whether the server will be required to submit additional information to the client [172].

In terms of satisfying security attributes, the negotiated encryption algorithm provides data confidentiality, as it protects the transmitted data from being eavesdropped on [193]. The exact keys used for MAC, encryption, authentication, and the secure hash function are derived from the master secret that is obtained from the pre-master-secret [167, 172, 193].

Furthermore, the TLS record protocol, which is the lowest layer of the protocol [172], takes higher layer TLS control messages or application data blocks to be fragmented and compressed. This is followed by appending a MAC that is derived from the master secret [172, 193] to secure the compressed fragment [172] and to ensure message integrity [193]. Finally, symmetric encryption is performed and the TLS record header is added in order to be passed on to the TCP for further processing [43, 172]. Thus, it can be said that MAC and encryption represent the fundamental security measures of the TLS record protocol [172]. MAC also includes a sequence number that helps in detecting extra, missing, or repeated messages [172].

Established session keys are ephemeral and have limited crypto-periods within the message sequence space [167]. Therefore, allowing re-keying for long sessions is an appropriate solution that prevents cryptanalysis in the case that a key is compromised [172]. In the re-keying process, the MAC and encryption keys of both parties are derived from the master secret. This is followed by fragmenting the key block into four successive keys, representing the pending key pair to be used next. Re-keying takes place within the re-negotiation phase, whenever the Change Cipher Spec message is exchanged between participants. The Change Cipher Spec protocol indicates that the sender and recipient cipher suite is being updated [172, 191]. Furthermore, one of the main functions of the re-negotiation phase is to protect the client's identity. This is because sending the "optional" client's public key certificate upon server request during the initial unencrypted handshake causes the client's identity to be exposed in clear text. Therefore, client authentication via certificate (if requested) is performed within the second handshake following the initial phase [172].

The finishing phase provides a mechanism for checking the integrity of all previous messages and exchanged parameters. The finished messages are based on the use of the secret master key, along with the application of a hash function to all the previous messages of the current TLS handshake, to generate the data to be verified. After validating the finished messages, transmission might continue or be aborted [172].

2.7.1.1 TLS/SSL Session Key Establishment

According to Section 2.5.1, secret key distribution is one of symmetric cryptography's shortcomings [153, 157, 160], but it can be solved with public key protocols [157]. RSA and the Diffie-Hellman key exchange (DHKE) are examples of such protocols [157, 171]. They are based on exchanging or generating the parameters used to generate the secret session key [171, 172]. The exact parameters used to generate the secret session key are determined by the key exchange algorithm [171] that was agreed upon during the initial handshake, which in turn defines the exact functionality of the TLS protocol [172].

As TLS usually uses a public key certificate for authentication [194], the server sends its certificate to the client following the initial negotiation to support the key exchange protocol [171, 172]. The certificate key type should be compatible with the negotiated key exchange algorithm. Hence, either the RSA/DSS or DH public key is used [171, 172].

2.7.1.1.1 RSA Key Exchange Method

Agreeing upon RSA as the key exchange algorithm within the initial handshake negotiation results in the use of an RSA public key as the server's certificate key type during the key exchange mechanism [171, 172]. The RSA key exchange method authenticates the server to the client by sending the public key certificate to support the key exchange protocol [172]. Following the certificate's verification, a pre-master secret (PMS) is generated by the client [172, 195] using the following equation:

$$PMS \coloneqq (Vers_C, RANDOM[46]) \qquad [172]$$

Where $Vers_C$ is the client's suggested version of TLS, and *RANDOM* [n] is a function that returns n unified random bytes [172].

PMS is considered the seed from which both connecting parties generate the secret session key (master secret) [172, 195]. In order to protect the confidentiality of the session, the PMS must remain protected. Therefore, the client encrypts the PMS with the server's public key and securely shares it with the server [172, 195]. Finally, both sides of the TLS protocol employ a pseudo-random function (PRF) that uses the exchanged parameters of the TLS initial handshake and the pre-master secret to generate the master secret (MS) [171, 172], using the following equation:

$$MS := PRF_{48}(PMS, \text{"master secret"}, (R_C, R_S)) \qquad [172]$$

The MS is then used for encrypting/decrypting the transferred messages over the negotiated session [171, 172, 195]. It is assumed that the encrypted PMS in the RSA key exchange method cannot be determined by eavesdropping, as it can only be decrypted by the server's private key [195]. Figure 2.8 shows the exact steps involved in the RSA key exchange method within TLS.

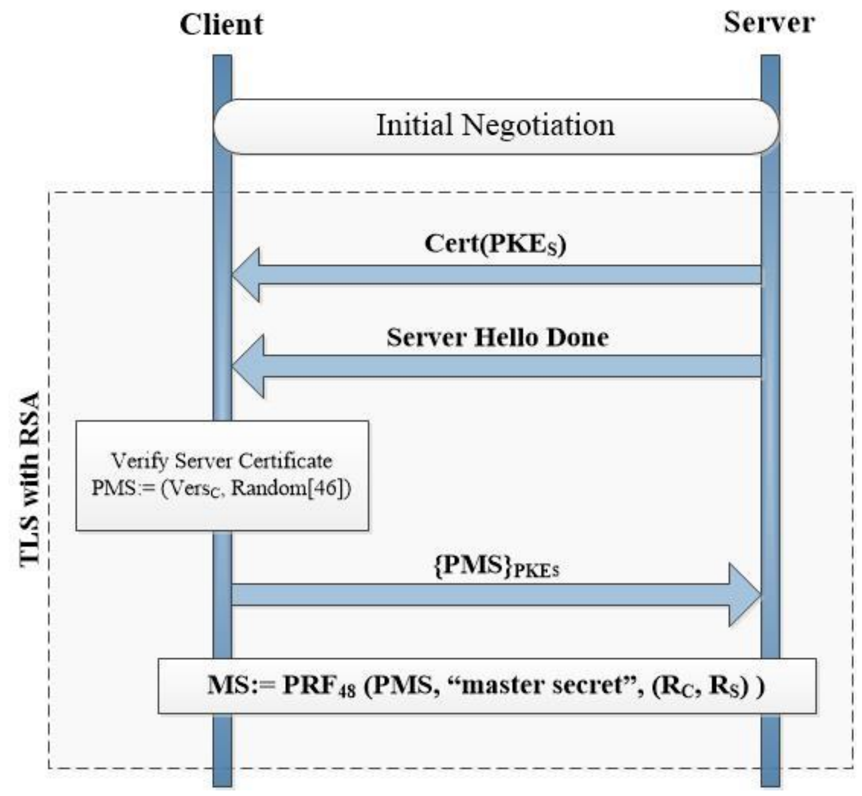
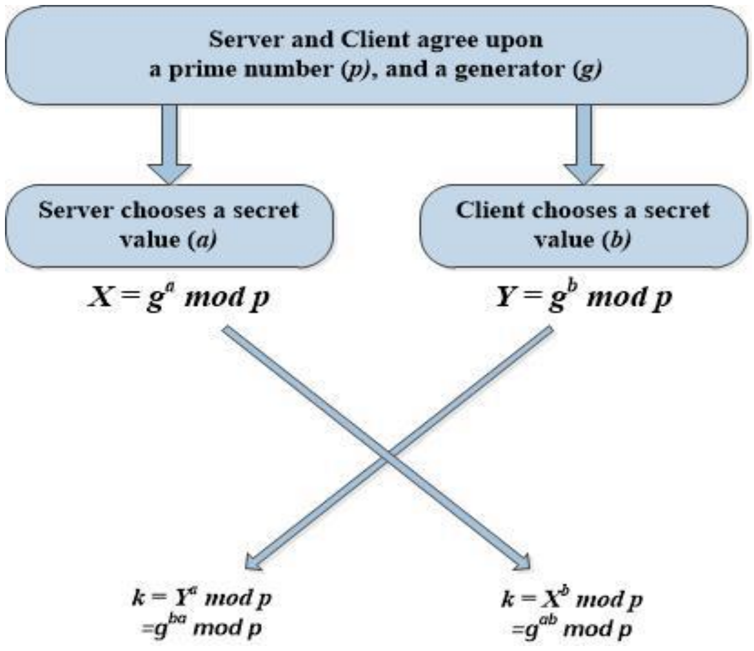

Client
Server
Initial Negotiation
TLS with RSA
Cert(PKE$_S$)
Server Hello Done
Verify Server Certificate
PMS:= (Vers$_C$, Random[46])
{PMS}$_{PKE_S}$
MS:= PRF$_{48}$ (PMS, "master secret", (R$_C$, R$_S$))

Figure 2.8: TLS with RSA Key Exchange Method

2.7.1.1.2 Diffie-Hellman Key Exchange (DHKE) Method

The Diffie-Hellman key exchange method (DHKE) is a fundamental key exchange technique that was first published in 1976 in Diffie and Hellman's landmark paper [168]. It offered the first effective solution to the secret-key distribution problem [153]. DHKE is an asymmetric cryptographic protocol that allows two parties communicating over an insecure channel to establish a shared secret key without the key being transmitted [153, 157, 196]. Thus, it is widely used in the key exchange frameworks of SSL/TLS and IPSec protocols as a means to ensure web security [157, 160].

The DH algorithm is composed of two phases: the setup phase and the key exchange phase [157]. The setup phase is responsible for publishing the public domain parameters (a prime modulus p and a generator g), which are used by both parties in the key exchange phase to compute their public values. The public values are exchanged between parties over an insecure channel to finally compute the shared session key without ever being transmitted [153, 157, 160, 196]. Figure 2.9 shows the mathematical computations of the DH algorithm.



Server and Client agree upon a prime number (p), and a generator (g)
Server chooses a secret value (a)
Client chooses a secret value (b)
$X = g^a \bmod p$
$Y = g^b \bmod p$
$k = Y^a \bmod p$
$= g^{ba} \bmod p$
$k = X^b \bmod p$
$= g^{ab} \bmod p$

Figure 2.9: Diffie-Hellman Mathematical Computations

In terms of security, an attacker already knows the domain parameters published during the setup phase, and there is also the possibility of easily obtaining the computed values exchanged between parties over insecure channels. Thus, the question arises of whether the attacker is capable of computing the shared session key from the compromised parameters. This is known as the Diffie-Hellman problem (DHP) [157].

Although it hasn't been mathematically proven, it is often assumed that efficiently solving the Discrete Logarithm Problem (DLP) in DH key exchange is the only efficient way to solve the Diffie-Hellman problem (DHP) [157]. Therefore, it is recommended to choose a sufficiently large prime number as the modulus, ensuring that the corresponding DLP cannot be solved [157, 160], as it becomes practically infeasible to compute the shared session key

[153]. Furthermore, the DH key exchange method supports anonymous cipher-suites (DH_ANON), which require no certificate authentication from either side and exchange unsigned DH parameters [172, 197]. This algorithm thus makes the system vulnerable to man-in-the-middle attacks (MITM) [197].

Selecting the DH exchange algorithm during the TLS initial handshake protocol results in server authentication using the DH public key certificate, which is digitally signed with either the RSA or DSS public key [171, 172]. The DH key exchange method is quite similar to the RSA key exchange method, except that it employs the DH algorithm [172, 194] to generate and exchange the secret session key [172], along with using the following equation:

$$MS \coloneqq PRF_{48}(g^{ab}\ mod\ p, \text{"master secret"}, (R_C, R_S)) \qquad [172]$$

Figure 2.10 shows the exact steps involved in the DH key exchange (DHKE) method within TLS.

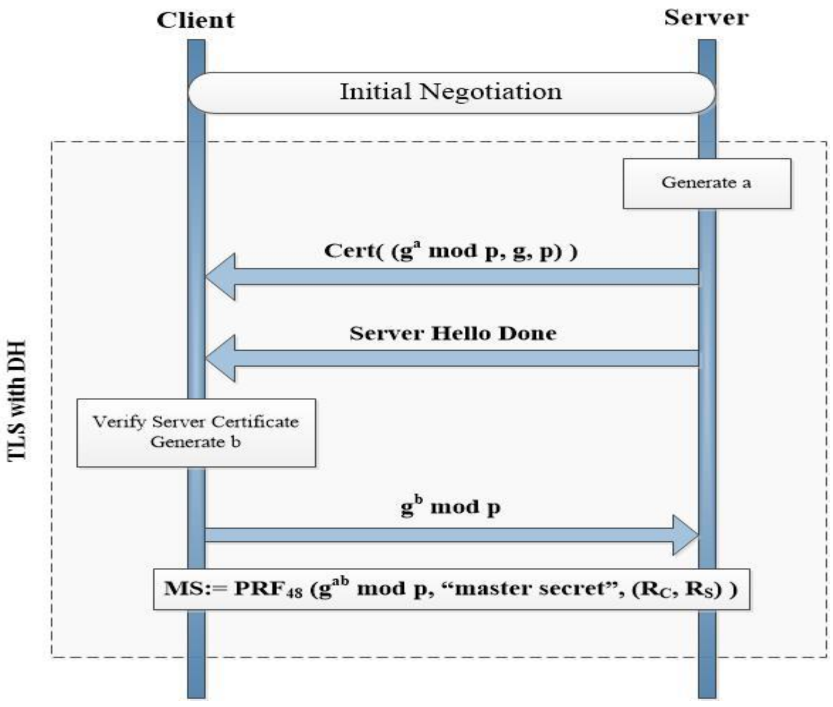

Client
Server
Initial Negotiation
TLS with DH
Generate a
Cert((g^a mod p, g, p))
Server Hello Done
Verify Server Certificate
Generate b
g^b mod p
MS:= PRF48 (g^ab mod p, "master secret", (RC, RS))

Figure 2.10: TLS with Diffie-Hellman Key Exchange Method

According to Huang et al. [197], the server's private keys can be compromised by attackers. Thus, RSA is not the optimal method for exchanging keys. An attacker intercepting traffic during the handshake is able to decrypt the transferred pre-master secret (PMS) to derive the shared master secret (MS), which is an essential security parameter in the re-negotiation and finishing phases within the TLS/SSL protocol. On the other hand, the PMS is not employed within the DH key exchange algorithm; the shared MS is calculated on both sides of the connection based on the exchanged public values [172, 195]. As the parameters used to derive the shared MS are not transferred through the network, an attacker intercepting the traffic cannot compromise the shared master secret.

2.8 Related Work

Regarding the proposed objective of this thesis, which is overcoming man-in-the-browser (MITB) attacks within an online authentication system, several related solutions have been proposed. Nor et al.[4] proposed a scheme that is based on incorporating a TPM-based remote attestation in order to deliver a trust communication between the client and the server that will mitigate an MITB attack. The protocol is based on the client measuring the platform integrity and storing it in the platform configuration registers (PCR) to be used in the registration phase. The protocol also uses the secure remote password (SRP) in the key exchange phase to satisfy a zero-knowledge proof requirement. In the final platform attestation phase, the client signs the PCR values with the AIK private key. The signature is then encrypted with a shared key and sent to the server for integrity verification. The server decrypts the received message to verify the signature, using the AIK public key that was previously sent during the registration phase. Once the signature is verified, both parties start their communication over a secure channel.

The experimental results have shown that the PCR values have changed, indicating that the integrity of the client platform has been tampered with. Hence, the protocol rejects the authentication. In general, moving security to the hardware level along with software provides more protection. However, TPM secures hardware against its owner through abusive remote attestation. In addition, the cost and size overheads of TPM chips are unacceptable [198, 199].Moreover, in case in which a system has been modified by an attacker who has administrative access to the system, as the TPM authenticates a system instead of the user of the system, the TPM cannot be reliable when it comes to security.
Goyal et al. [3] proposed an image-based approach to mitigate MITB attacks during online banking. The approach is based on the bank server embedding critical transaction

information, submitted by the user on a user-specific personal image (USPI) that has been selected earlier by the user via a bank visit. Next, an HTML confirmation page displays the transaction information, along with the embedded USPI, for the user to confirm the details and to detect any irregularity of the information displayed on the USPI before submission. Once a change has been detected, the user cancels the transaction. The approach encompasses several countermeasures against the attacker, such as inaccessibility of the USPI, and the inability to find an automated algorithm that can extract and change the information displayed on the USPI within the bank confirmation time limit, along with eliminating any USPI artefacts. On the other hand, the MITB can confirm the transaction without the user getting a chance to notice any irregularity within the USPI. Thus, Krishnan and Kumar [12] proposed a solution that requires the user to submit an OTP that is embedded within the USPI to confirm the transaction. However, the approach lacks practicality due to the need of the user visiting the bank. In addition, the protocol lacks decision making as it depends on the user to either discard the process or to carry on with it.

In a recent and similar user authentication approach against MITM and MITB attacks, Chow et al. [200] proposed a two-phased scheme, an initial user authentication phase, and a transaction verification phase. The user authentication phase initiates the scheme by submitting a user ID via an untrusted computer for public-key retrieval from a PKI. A cryptographic random number-based OTP is generated, encrypted with the user public key, and embedded within a QR code. The OTP is decrypted on the mobile phone side to be retrieved and submitted to the server for verification via the untrusted computer. Next, the transaction verification phase requires transaction data submission via an untrusted computer. Similarly, a random number-based TAN is generated, symmetrically encrypted, and embedded within a QR code, along with the transaction data and a hash of the transaction data, credential, and the encrypted TAN. The mobile side verifies the recovered hash, and decrypts the TAN to be submitted via the untrusted computer after checking the transaction data. Although the scheme provides a strong authentication and transaction system against MITM and MITB attacks, in which any attempt to manipulate the transaction data or reuse the OTP/TAN will fail, both phases still use the untrusted browser for submitting the OTP and the TAN. Thus, the user can abort the whole process in case the transaction data have been manipulated, whereas the proposed MUAS has transitioned the whole authentication process to a new and detached connection with no risk of MITB attack and no need to abort the process. Moreover, in case the user authentication phase is used as a stand-alone system, the attacker can simply wait for the user to become authenticated and then take over the process.

Moreover, several similar approaches have proposed employing QR code technology within online authentication systems to prevent MITM attacks. However, due to its similarity with the proposed MUAS, these approaches were studied in-depth regarding overcoming MITB attacks. Dodson et al. [201] proposed and implemented a strong and effective QR-based challenge/response approach called 'Snap2Pass' that has been made available to over 30,000 OpenID websites. This approach is implemented for OpenID-based authentication systems to overcome the problem of having too many usernames and passwords to memorise. Snap2Pass is composed of two phases, an account creation and account login, which are based on challenge and challenge/response generation, respectively. Whenever the account creation phase is initiated and the user submits a chosen username, a shared secret is computed by the server and encoded within the QR code, along with the provider name and response end point. Once the QR code is scanned, this secret is saved in the mobile phone password manager to be used for subsequent logins. On the login page, a challenging QR code, containing the browser nonce and the provider name will be displayed for the user to capture. Once the QR code is scanned and decoded, the application will use the recovered provider name as a reference to find the shared secret and the end point on the mobile. Once found, the response is generated using the hash-based MAC along with the shared secret to be sent to the end point, along with the original challenge and the user name. No input is required during the login phase, thus providing strong security against an MITM attack. However, an MITB attack can simply wait for the user to become authenticated and take over the transaction via the untrusted browser. Moreover, the system lacks user legitimacy verification due to not requiring the certificate authority (CA) infrastructure in both phases. In addition, the account creation phase displays an unencrypted QR code, where a peeper can capture the displayed QR code and log in using the victim's identity, unlike the proposed MITB-resistant MUAS, where both phases employ PKI for mutual identity verification and encrypting the QR code content, using the client's CA-issued SSL-certificate.

A similar QR-based challenge/response approach called '2-clickAuth' was proposed by Vapen et al.[130]. The proposed approach implemented a QR-based identity provider, making it available to users of OpenID websites, including Facebook, SourceForge, and

MySpace. It replaced the need to submit a password within the OpenID provider. Following the submission of the OpenID within the relying party and being redirected to the OpenID provider, the proposed 2-clickAuth starts its authentication process. A secret is generated by the OpenID provider and encoded into a QR code, which is captured by the mobile phone to generate the response using the hash-based MAC along with a shared secret key between the mobile and OpenID provider. The generated response is then encoded into a QR code, which is captured by the web camera of the untrusted computer to be sent to the OpenID provider for verification. If it verifies as true, the user is authenticated and redirected to the relaying party. As a requirement for the proposed approach, the mobile phone needs to register with the OpenID provider. Therefore, the mobile phone encodes an encrypted freshly generated secret key into a QR code to be captured by the web camera of the untrusted browser. Consequently, the mobile phone and the OpenID provider share a secret key. The '2-clickAuth' approach is secure enough against replay attacks and MITM and MITB attacks, as it only employs the optical channel in the key exchange mechanism and in the authentication mechanism, back and forth. However, the system lacks practicality in terms of the response sending technique within the authentication system.

From the above literature, the proposed MUAS protocol and [200] employ the public-key infrastructure for identifying and verifying the identity of the participants. In addition, QR code contents are encrypted via public key encryption, where the mobile phone application have access to the client's private key, unlike Snap2Pass [201] and 2-clickAuth [130] that lacks the confidentiality of the QR code. Furthermore, the protocol in [200] aborts the authentication mechanism in case of an MITB attack. Unlike the proposed MUAS, which carries on with the authentication process using a different communication channel.

2.9 Conclusion

In this chapter, we summarise the various components involved in developing a secure online authentication system. The individual components are combined to produce an e-commerce security system and its dimensions. E-commerce security systems are generally composed of five layers: the network service layer, encryption technology layer, security authentication layer, security protocol layer and application layer. The network service layer is responsible for providing the e-commerce security system the ability to solve network problems. The remaining layers represent basic building blocks for e-commerce transaction security, which are cautiously adopted to help develop an e-commerce authentication system.

Next, an in-depth study was conducted to test and evaluate three fundamental authentication mechanisms: the single-factor, two-factor and three-factor authentication. These mechanisms were critically reviewed according to their associated advantages and disadvantages. The end-result of this analysis determined the most suitable technique, which does not satisfy the objectives of this thesis. In addition, a comparative study focusing on 11 popular UK banks was conducted to determine and evaluate state-of-the-art identification and authentication mechanisms. This study reveals that two-factor authentication (2FA) methods are the authentication mechanisms most frequently used by financial institutions; this is because they are considered the most suitable mechanisms to secure the logging-in process of their online banking platform. As 2FA is based upon object possession. Many technologies have emerged to exploit such objects; these include authentication mechanisms such as one-time-passwords (OTP) and quick response codes (QR code). Although OTP are considered one of the strongest authentication mechanisms in use today, along with static passwords, it cannot accomplish the objective of preventing or surmounting an MITB attack. The mechanism can instead be used for verifying a mobile phone number in case of submission.
Aside from OTP mechanisms, another technology that exploits the most popular and available authentication device was considered; this technology is the popular QR code, which requires a visual channel for capturing and decoding the QR code. As a result, mobile phones can become the perfect authentication device to use with QR code due to its expanded functionalities. These functionalities include downloading applications for code decoding and sending/receiving messages in case an OTP mechanism was involved. Moreover, the connectivity feature of mobile phones is another opportunity that can be exploited within the process of developing a secure authentication system, due to its ability to connect to the internet. Thus, QR code technology can be employed to transition the authentication process from an MITB-infected browser to a more secure mobile phone.

Determining the proper technologies that can defeat an MITB attack is considered the first step in establishing e-commerce transaction security layers. This process suggests that each layer has several technologies that can enhance the security mechanism. Accordingly, the appropriate technology is chosen to serve the purpose of the proposed system at each layer.

At the encryption technology layer, asymmetric cryptography is chosen as a way of enhancing the security of the proposed MUAS by encrypting the QR code to prevent unauthorised information from being disclosed to unauthorised observed. While the security authentication layer is responsible for verifying the identities of participants within the e-commerce platform, several technologies were employed for mutual authentication, including digital certificates and certification authority, which are components of a PKI. In addition, cryptographic nonce and digital digest are employed to prevent replay attacks and generate a message digest of the confidential information, respectively. Finally, the SSL/TLS protocol was chosen within the security protocol layer to establish a secure communication channel between participants, thus protecting all exchanged messages.

3 Design and Methodology

3.1 Introduction

A review of the literature and online authentication approaches have been presented in the previous chapter, paving and justifying the way to detail the different aspects involved in designing the methodology of the proposed online Mobile-User-Authentication System (MUAS).

The infrastructure of an online authentication system represents the foundation on which a secure system is built upon. Thus, employing the Public Key Infrastructure (PKI) along with the SSL/TLS protocol provides the basis for building a strong and secure authentication system. In terms of a user authentication session, a novel Mobile-User-Authentication System (MUAS) is proposed. MUAS employs SSL/TLS protocol within e-commerce applications as a security backbone to essentially authenticate the server via its PKI-based SSL-certificate, and to secure all exchanged messages among participants [202].

Based on the proposed objective, MUAS design is based on developing a novel challenge-response protocol that overcomes Man-In-The-Browser (MITB) attack. Therefore, the proposed approach is divided into two physically detached and logically connected phases, which are the challenge- and response-generation phases. When designing the proposed system, security properties were taken into consideration alongside the primary objective. Thus, cryptographic security attributes [41] were utilised within exchanged messages for security purposes and for the logical linkage between both phases. In addition, MUAS registration information were utilised within both phases for preventing DOS attack and ensuring the authenticity of the primary authentication device.

The proposed authentication scheme can be the basis for conducting secure online transactions, where the authentication is initiated within the infected browser and the verification of the identity and the transaction are performed on a detached medium. Taking MITB capabilities into account, the initiation phase of the proposed system limit the amount of the requested information from the user to be composed of the mobile number only. Thus, stealing mobile numbers and choosing targets that requests non-confidential information are not of interest to the attacker. Moreover, modifying and tampering with outgoing mobile numbers will only be an act of vandalism with no effect on the client. Finally, modifying HTML pages within the context of the proposed scheme will involve adding extra data fields that request financial information, whereas the client is only registered through the mobile number and the IMEI for being authenticated within the site and not for any financial transaction.

This chapter detail and justifies the different parts and aspects of the design, along with the main goal of this thesis, and is organised into the following sections. Section 3.2 provides an overview of the proposed MUAS design model, along with its entities and basic architecture.

Plus, the design assumption on which the MUAS is built are also provided. Section 3.3 present the objectives and security requirements of the challenge generation phase, along with an abstract overview of the whole phase. In addition, a justification of the QR code design issues are presented and explained, as QR code generation process represents the foundation behind the logical connection between the two phases. Section 3.4 present the objectives and security requirements of the response generation phase, along with an abstract overview of the whole phase and its layers. Section 3.5 concludes the different design aspects of this chapter.

3.2 MUAS Design Model

Based on the attacker's interest in financial information, once the client is authenticated and starts providing confidential information, the attacker starts its malicious act. Thus, conducting a secure authentication represents the preliminary step for conducting a secure transaction. Therefore, the aim of this thesis is to design an online cryptographic-based Mobile-User-Authentication System (MUAS) for e-commerce websites. With respect to avoiding MITB attack, the proposed system authenticates clients securely through transitioning the authentication mechanism from the infected browser to a new and detached medium, which utilises the mobile phone as the primary authentication device. . Hence, the proposed MUAS has been developed under the foundation of a challenge/response protocol, which corresponds to two connections that are physically separated, yet logically related. Therefore, the MUAS is split into **two phases**:

- **Challenge generation phase**,
- and **Response generation phase**.

The MUAS is based on establishing a platform on which the client can interact and communicate smoothly with the server to generate a cryptographic QR code challenge, for the client to capture and to generate the response. Therefore, the platform is considered a prerequisite for the challenge- and response-generation phases, for information submission, challenge generation, and response verification processes. However, before the MUAS mechanism begins, the client should be registered through submitting the mobile number and the International Mobile Equipment Identity IMEI of his/her mobile phone to be saved in the $\boldsymbol{Registered - clients\,Table}$. As seen in figure 3.1.

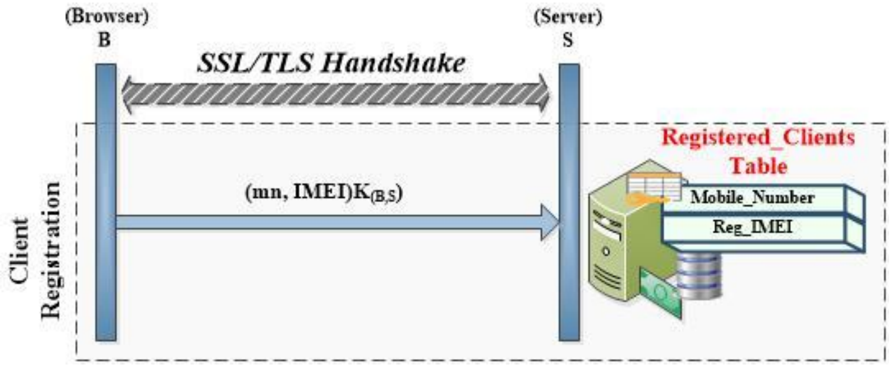


Figure 3.1: MUAS Client Registration

The system consists of the following entities: an **end-user** and an **authentication server**. The end-user is equipped with a personal computer and a mobile phone device with a network connection, assigned to the challenge- and response- generation phases, respectively.

Both PC and mobile phone share the same PKI X.509 certificate, required for establishing an SSL/TLS communication channel between participants. Although certificates are capable of verifying identities, however MITB attacks can bypass the security measures of public key infrastructure (PKI) and SSL/TLS protocols [14]. Consequently, using PKI X.509 certificates alone within infected browsers for authentication purposes cannot be held accountable. Therefore, a challenge that switch the client from the infected browser to the mobile phone is required for authenticating the client within the mobile phone side.

The authentication server is considered as the MUAS platform, where it is being accessed by the user in both phases for online authentication. Furthermore, a visual channel takes place within the response generation phase through employing a camera-based mobile phone, for communicating with the browser via QR code capturing. At a high level, the proposed MUAS works as follows. The client initiates a connection with an e-commerce website. The authentication server of the website requests the client to submit his/her mobile number. Upon receiving a registered mobile number, the server generates a one-time-password (OTP) and submits it to the mobile phone via SMS. Next, the authentication server request the client to submit the OTP. Following the OTP verification, the authentication server generates an encrypted QR code that encodes a cryptographic challenge, along with other contents. The client captures the QR code with the mobile phone camera. The mobile phone computes the response from the cryptographic challenge and sends it to the authentication server. The server verifies the response, and if it verifies to true, the client is authenticated. Under the hood, security attributes are exchanged between participants. Figure 3.2 is a high level representation of the proposed MUAS.

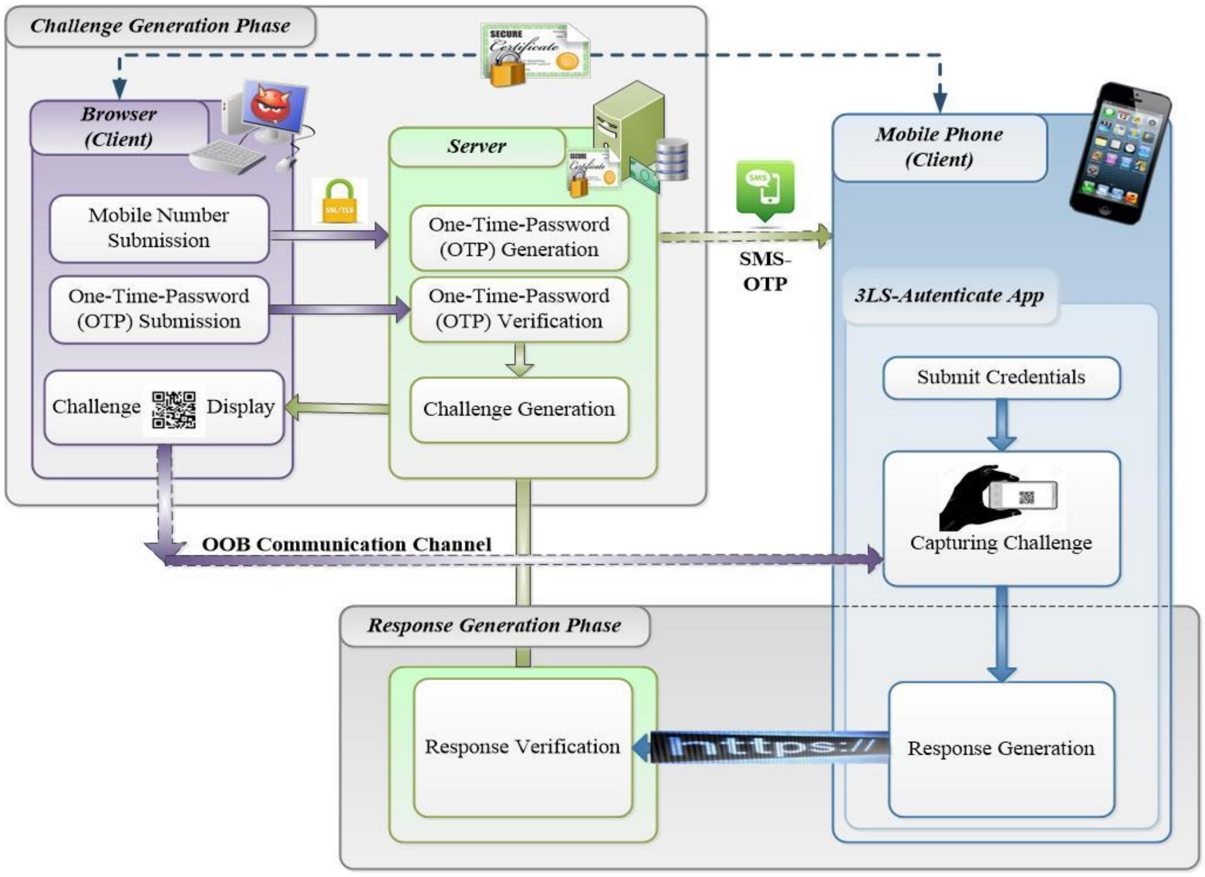


Figure 3.2: MUAS High Level Representation

3.3 MUAS Design Assumptions

When designing the proposed protocol, a number of assumptions were taken into consideration:

- Both client and server are pre-registered with a trustworthy Registration Authority (RA). The RA enters into an agreement with the Certification Authority (CA), for issuing a trusted and signed X.509 certificate for the applicant. The issued certificate contains a public key, along with a number of attributes, some of which forms an identity. The identity is included in all exchanged messages within the proposed protocol, which helps obtain the public key of the partner from the PKI.

- The PKI X.509 certificate of the client is shared between the PC and the mobile phone without any modification of the existing standard format. This assumption was based on the certificate sharing system (CSS) proposed by Kim et al. [184], which provides an extensible and resilient system for distributing certificates within an enterprise environment. The X.509 certificate, which is saved on the PC as a .CERT file is converted into a .PFX file after authenticating the user via user ID, password, and certificate password. The .PFX file is then converted into a binary code and stored on the database server. On the mobile phone side, the user downloads the binary code from the database server after being authenticated via the user ID, password, certificate password, and OTP. The binary code is then converted into a .PFX file, followed by conversion into a .CERT file to eventually be saved on the mobile phone. The X.509 certificate stored on the PC is the same as the X.509 certificate stored on the smart phone, thus ensuring the correctness of the digital certificate installed on the mobile phone. This approach was discussed in section 2.6.3.

- User identities exchanged between parties in both protocols are constructed from the Distinguished Name (DN) field values [203-205] of the issued X.509 certificate. The user ID can be either the subject name or the issuer name along with the serial number of the certificate [206].

- An SSL/TLS communication channel is established between both participants in both protocols as a security backbone and the fundamental cryptographic protocol within the context of a client-server model, thus establishing shared secret keys for securing all exchanged messages within both protocols. This reliable end-to-end secure channel is used for authenticating the server via its X.509 certificate.

- Shared session keys in both protocols cannot be compromised. Compromising such keys is beyond the scope of this thesis, as they represent the result of the negotiation phase within the SSL/TLS handshake protocol [172]. Although, it is compromised in the challenge generation protocol as the result of applying the MITB attack model, which is discussed in the next chapter.

- The server of the proposed MUAS requires prior client registration only through the mobile number and the IMEI of the mobile phone. Thus, the challenge generation phase requires mutual authentication of both client (browser) and server within the SSL/TLS handshake protocol via their X.509 certificates for security purposes. Whereas the response generation phase requires server authentication only, as the mobile phone represents the same client in the challenge generation phase, thus they share the same SSL-certificate.

3.4 Challenge Generation Phase

The challenge generation is the first phase of the proposed Mobile-User-Authentication-System (MUAS), which is responsible for initiating the whole authentication process. This phase is based on the client initiating a connection with a merchant through his/her browser. The challenge generation phase takes place following the SSL/TLS handshake protocol establishment, which resulted in mutual authentication between participants, thus sharing a jointly secret key for securing exchanged messages. The browser initiates the authentication system through exchanging and verifying cryptographic messages with the authentication server as part of securing the protocol. The exchanged messages are composed of the identity of the sender, freshly generated cryptographic nonce, along with the submitted information from the client, upon request.

From a design point of view, the MUAS is designed as a user-friendly and non-confidential input-based authentication system that mitigates the denial-of-service (DOS) attack and overcomes the MITB attack. The system only request the client to submit the registered mobile number within the infected browser as a means to generate a challenge that transitions the authentication mechanism to the mobile phone of the client, which can be used for validating the identity of the client along with conducting the transaction without the risk of MITB attack. Upon mobile number validation, MUAS employs one-time-password (OTP) as an auxiliary authentication mechanism to ensure that the mobile phone is in possession by the client who initiated the authentication process. Hence, the main objective of the OTP authentication mechanism is to link the client to the submitted mobile number.

In the challenge-generation phase, whenever a registered client initiates a connection with the authentication server, a record for that client is created and added to the $\boldsymbol{Client - Table}$. The primary key of the table is the user ID of the initiating client $C\_\boldsymbol{UserID}$, which is included in all received messages. Alongside the primary key, the received mobile number $\boldsymbol{mn}$ will be saved in the $\boldsymbol{Mobile\_Number}$ field as the foreign key. The value of the fresh server-nonce $\boldsymbol{ns}$, which was uniquely generated by the server for the session will be saved in the $\boldsymbol{Server - nonce}$ field. Furthermore, other fields such as $\boldsymbol{One\_Time\_Password}$, $\boldsymbol{Server\_nonce2}$, $\boldsymbol{Mobile\_nonce}$, $\boldsymbol{Mobile\_Result}$, and $\boldsymbol{IMEI}$ are also included in the table. Figure 3.3 shows the relational database that contains the $\boldsymbol{Registered - Clients\ Table}$ and the $\boldsymbol{Client - Table}$.

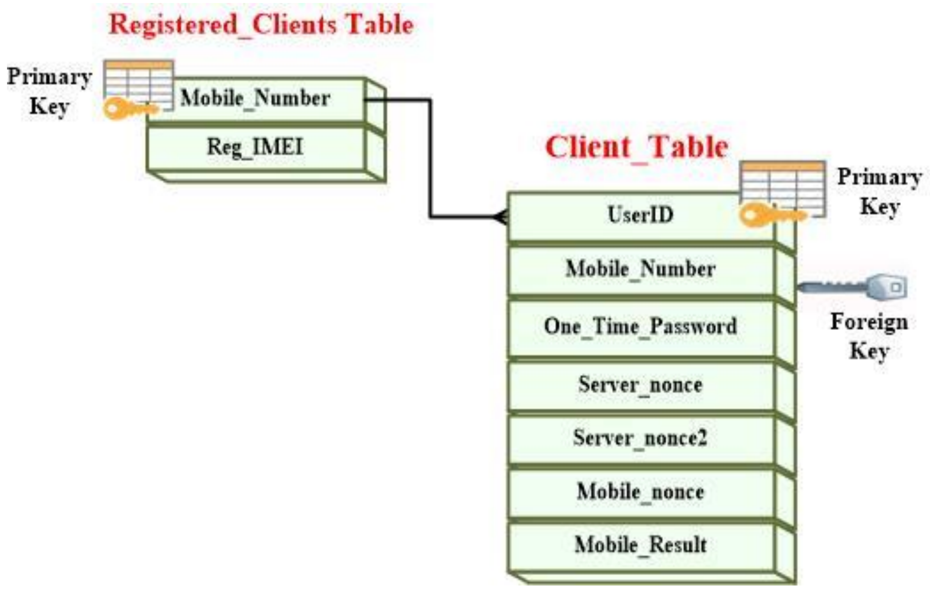


Figure 3.3: MUAS Relational Database

Following the submission and verification of the received SMS-based OTP, the server compose and append QR code contents to finally generate the cryptographic nonce-based QR code image. Figure 3.4 shows the high level representation of the challenge generation phase.

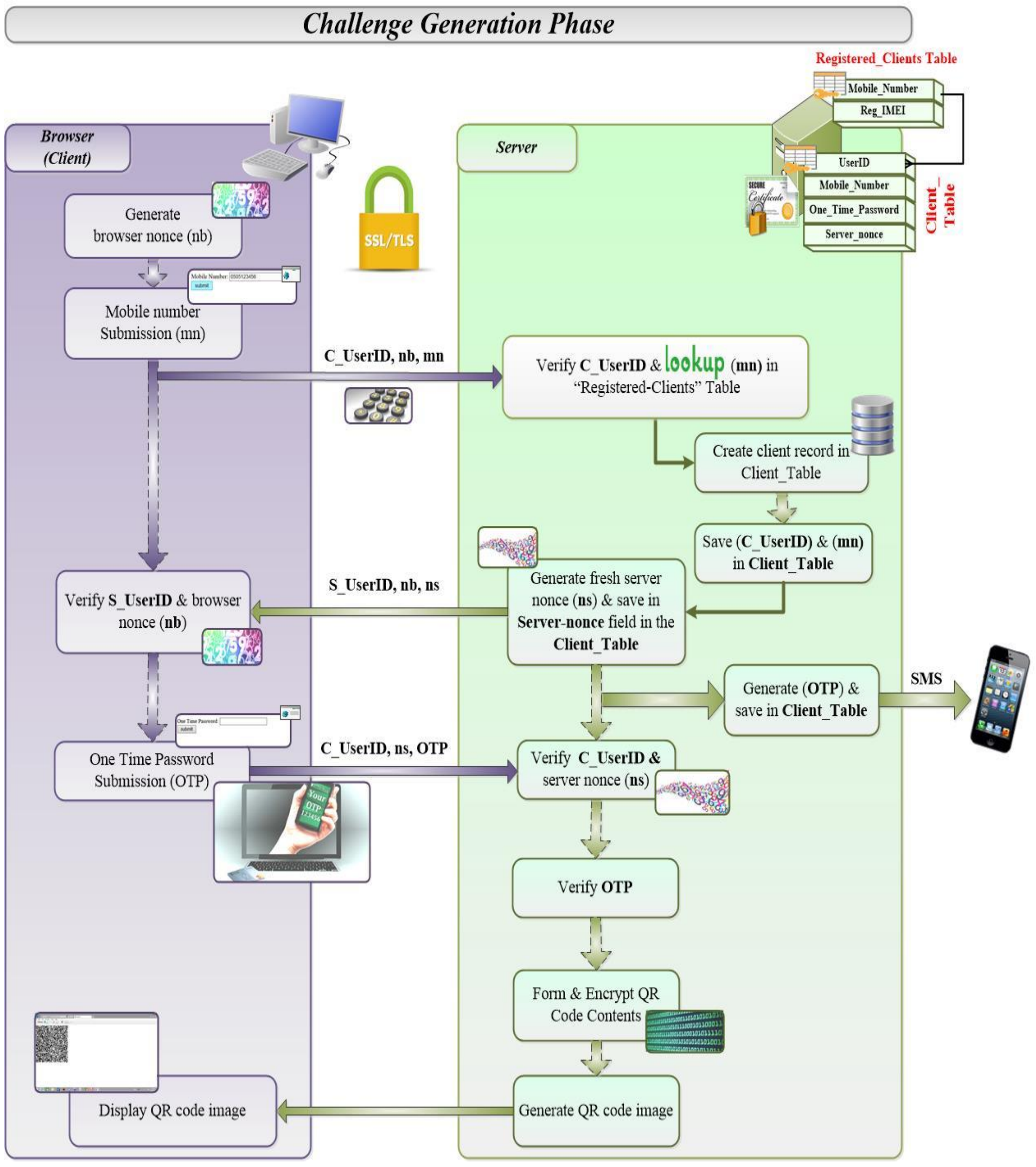

Challenge Generation Phase
Registered_Clients Table
Mobile_Number
Reg_IMEI
Browser (Client)
Server
SSL/TLS
UserID
Mobile_Number
One_Time_Password
Server_nonce
Client_Table
Generate browser nonce (nb)
Mobile number Submission (mn)
C_UserID, nb, mn
Verify C_UserID & lookup (mn) in "Registered-Clients" Table
Create client record in Client_Table
Save (C_UserID) & (mn) in Client_Table
S_UserID, nb, ns
Verify S_UserID & browser nonce (nb)
Generate fresh server nonce (ns) & save in Server-nonce field in the Client_Table
Generate (OTP) & save in Client_Table
SMS
C_UserID, ns, OTP
One Time Password Submission (OTP)
Verify C_UserID & server nonce (ns)
Verify OTP
Form & Encrypt QR Code Contents
Display QR code image
Generate QR code image

Figure 3.4: High Level Representation of the Challenge Generation Phase

3.4.1QR Code Generation

For achieving a secure and solid Mobile-User-Authentication System (MUAS), both challenge- and response-generation phases should complement each other. The foundation behind this complement lies within generating a challenging QR code with specific contents as a prerequisite for establishing a logical connection between the two phases. Therefore, embedding the proper information within the QR code represents the basis behind the coherency of both MUAS phases.

The main objective of the challenge generation phase is generating the challenging QR code to be displayed on the web-browser, for the client to capture and generate the response within the response generation phase. The QR code generation process is based on assembling the required contents to generate the code. Thus, the process is composed of two phases:

- **Forming QR code contents**,
- and **Generating QR code image**.

Forming QR code contents is the first phase of the QR code generation process. It takes place within the server-side, following the submission of the client's mobile number and one-time-password (OTP) verification. This phase is a multi-level process that consists of contents assembly, appending, and encryption. Contents assembly involves including the server's identity $\boldsymbol{S\_UserID}$ from its SSL-certificate, and appending it along with the server's initial and unique freshly-generated nonce $\boldsymbol{ns}$, and response end-point $\boldsymbol{URL}$. Figure 3.5 shows the two phases of the QR code generation process.

From a design point of view, embedding the freshly server-nonce into the QR code as one of its contents satisfies the uniqueness property, as nonce is never repeated during a session. The unique server-nonce represents the essential element of the MUAS response verification process. Another QR code influential embedded element is the server's distinguished name, which plays a major role in verifying the identity of the server in the challenge generation phase against the server in the response generation phase. Thus, ensuring that the end user in both protocols are communicating with the same authentication server. A further attribute considered while creating QR code contents is the server's URL. This URL is exploited to establish a secure SSL/TLS communication channel between the mobile phone and the server, for sending the generated cryptographic response from the mobile phone to the server for verification.

After completing the QR code assembly and appending processes, the server obtains the public key of the client (browser) from the PKI to encrypt the appended contents.

Next, the second phase generates the QR code image to finally be displayed on the client-side (browser). The QR code is finally captured by the client's mobile-phone camera to be used in the response generation phase for authentication and verification purposes.

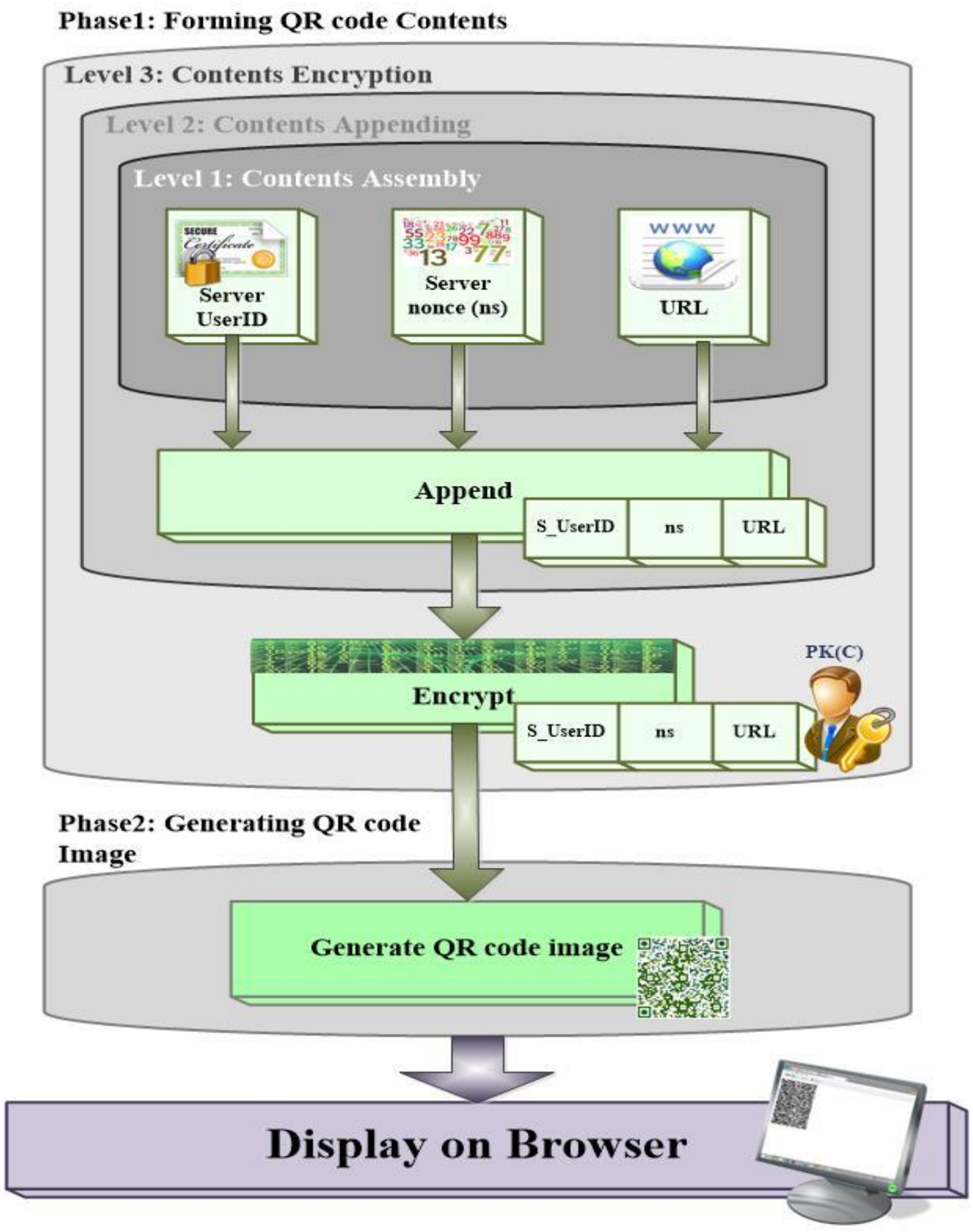

Phase1: Forming QR code Contents
Level 3: Contents Encryption
Level 2: Contents Appending
Level 1: Contents Assembly
Server UserID
Server nonce (ns)
URL
Append
S_UserID
ns
URL
Encrypt
S_UserID
ns
URL
PK(C)
Phase2: Generating QR code Image
Generate QR code image
Display on Browser

Figure 3.5: QR Code Generation Phases in MUAS

3.5 Response Generation Phase

The response generation is the second phase of the proposed Mobile-User-Authentication-System (MUAS), which is responsible for finalizing the authentication process by generating the response and sending it to the server for verification. This phase takes place between the mobile phone and the same authentication server platform, through a developed mobile phone application called "3LS-Authentication".

The "3LS-Authentication" stands for Three-Layer Secure Authentication. The application is composed of three sequential layers of security: activation, cryptography, and response generation. The application is responsible for capturing the generated QR code via the mobile phone camera, and generating a cryptographic response to be sent securely to the server for verification. Figure 3.6 shows the three sequential layers of the developed "3LS-Authentication" mobile application.

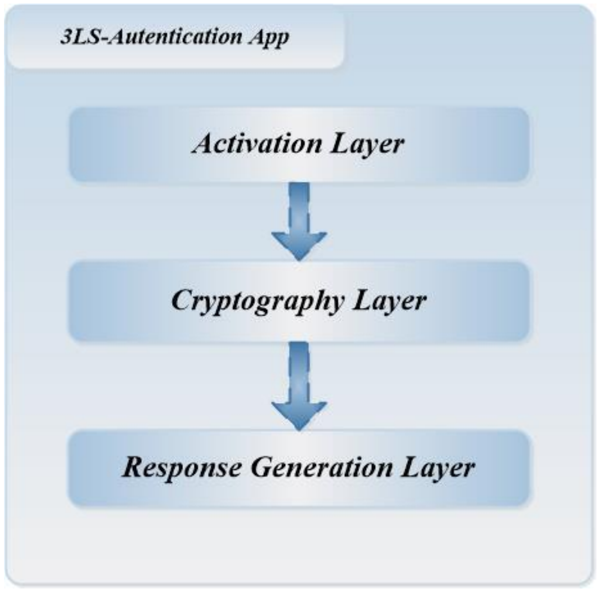


Figure 3.6: 3LS-Authentication Layers

Considering the SSL/TLS handshake's mutual authentication within the challenge generation phase, the response generation phase has been designed without requiring the SSL-certificate of the mobile phone, as they both represent the same client.

3.5.1Activation Layer

The activation layer is the first security layer of the application, residing in the client's mobile phone. This layer displays an alert view requesting the client to submit the login ID and password, as a requirement for the application to operate. Thus, allowing the client to capture the displayed QR code. Figure 3.7 shows the activation layer of the “3LS-Authneitcation” mobile application.

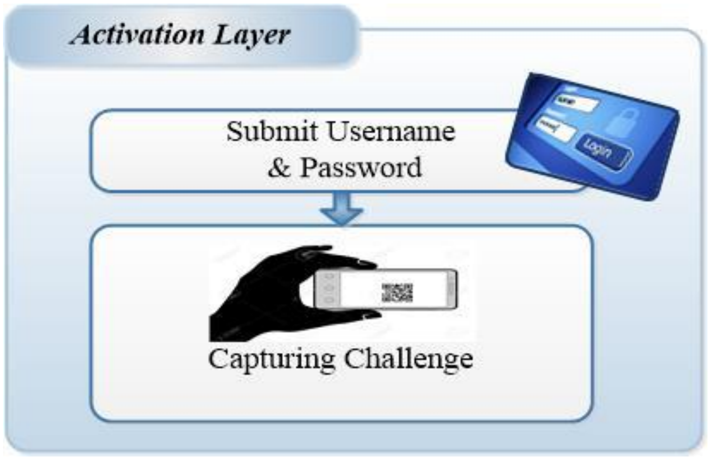


Figure 3.7: 3LS-Authentication Activation Layer

3.5.2Cryptography Layer

The cryptography layer is the second security layer of the application, which takes place following the QR code capturing. The foundation behind this layer lies within sharing the PC’s X.509 certificate with the mobile phone, from which the private key is utilised to decrypt the QR code as a means to recover, split, and exploit its contents. Moreover, the cryptography layer represents the basis for paving a cryptographic communication channel with the server for sending the response for verification.

The main objective of this layer is to utilise the functionalities of the QR code contents, for preparing an adequate environment for sending the response securely to the legitimate authentication server. In terms of establishing a secure communication session, the QR-recovered $\boldsymbol{URL}$ is exploited to establish a secure SSL/TLS communication channel with the server, for the mobile phone application to start exchanging and verifying cryptographic messages with the authentication server as part of securing the cryptographic protocol. Another objective lies within extracting the IMEI of the mobile phone to guarantee its legitimacy upon submission to the server. The IMEI is included within the initiating cryptographic message to the server, along with the extracted mobile number ($\boldsymbol{mn}$), mobile identity ($\boldsymbol{M}$), and the freshly generated cryptographic nonce ($\boldsymbol{n\_mobile}$). Upon receiving the first message, the server uses the received mobile number ($\boldsymbol{mn}$) to lookup the

$\boldsymbol{Registered\_Clients\ Table}$. If the mobile number was registered, the received IMEI is verified against the registered IMEI, thus ensuring the legitimacy of the mobile phone and mitigating the denial-of-service (DOS) attack.
Upon IMEI verification, the server generates a fresh and new server-nonce $\boldsymbol{ns2}$ for its session with the mobile phone. The mobile phone application verifies the identity of the server by validating the received server's identity $\boldsymbol{S\_UserID}$ against the QR-recovered identity $\boldsymbol{S\_UserID}$, and against the server's ID from the CA-issued SSL-certificate, received during the SSL/TLS establishment. Based on the server's authenticity and identity verification results, the final layer is executed and the response is generated.

3.5.3Response Generation Layer

The response generation layer is the third and final layer of the application, which takes place between the "3LS-Authentication" application and the authentication server. The foundation of this layer lies within exploiting the functionalities of the previous layer to securely generate and send the response to the server for verification.
With regard to response generation, the point of authenticating the server at the mobile phone-side represents the initiating point of generating the response. The server authentication point can be defined as the point of validating the mobile phone's cryptographic nonce $\boldsymbol{n\_mobile}$, exchanged with the server. Upon validation, the QR-recovered server nonce $\boldsymbol{ns}$ is hashed after being appended with the validated cryptographic nonce $\boldsymbol{n\_mobile}$, thus representing the generated response.
As part of the secure communication protocol, initiated in the previous layer between the mobile phone and the authentication server, the "3LS-Authentication" application embeds the generated response along with the exchanged cryptographic server-nonce $\boldsymbol{ns}$, and the identity of the sender $\boldsymbol{C\_UserID}$, and sends it to the server over the SSL/TLS connection for verification.

Finally, upon receiving the response from the "3LS-Authentication", the server verifies the identity of the sender against the listed clients within the $\boldsymbol{Client - Table}$. Once the client is found, the server extracts the content of the $\boldsymbol{Server - nonce}$ and the $\boldsymbol{Mobile - nonce}$ fields from the $\boldsymbol{Client - Table}$ and append them together. The server finally applies a hash function to the appended result to be compared against the received response. Consequentially, and based on the response verification result, the authentication result is sent to the mobile phone, either to reject the client or to carry on with a future process. Figure 3.8 shows the cryptography and response generation layers of the "3LS-Authentication" mobile application, along with the server verification.

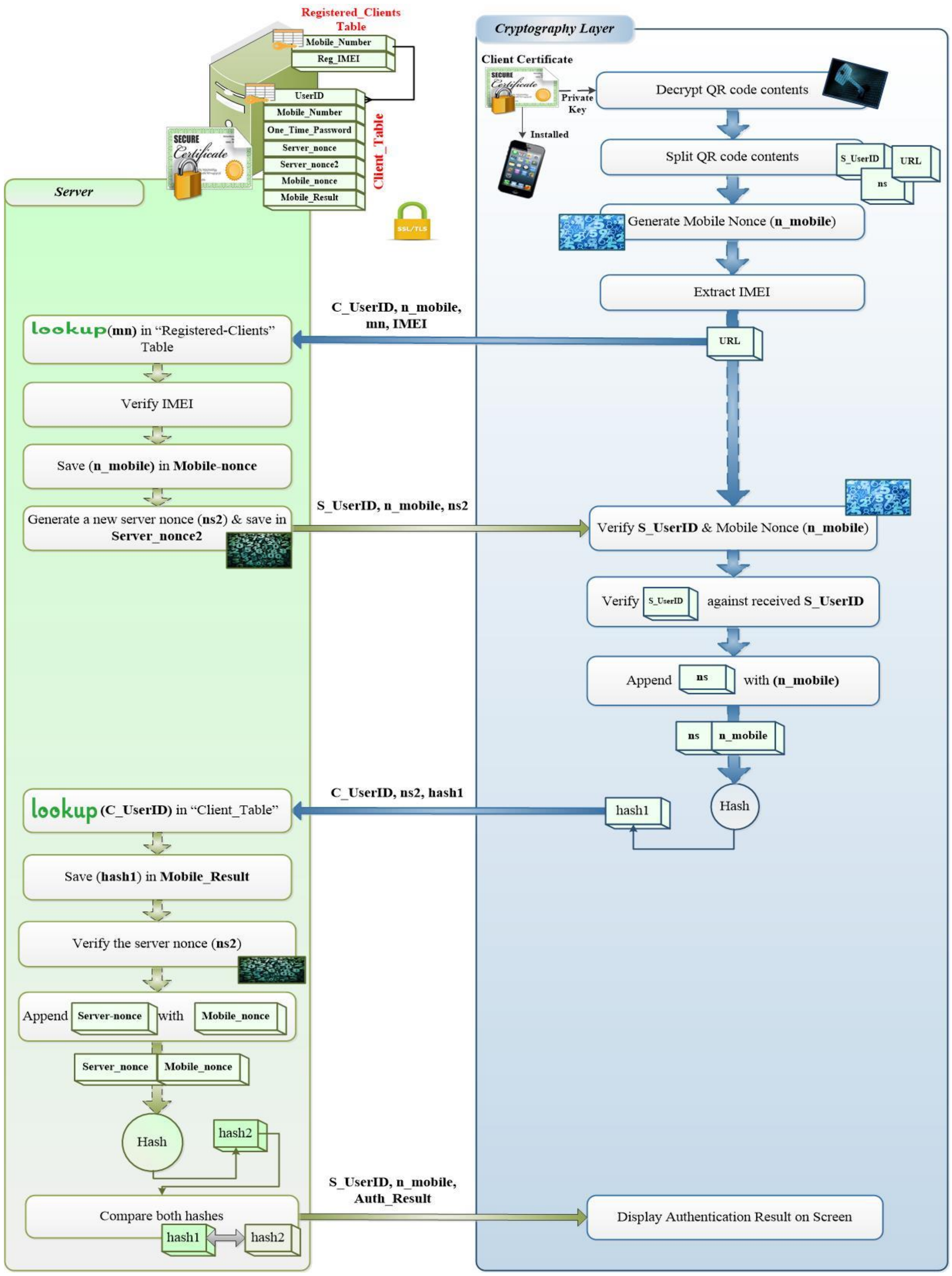

Registered_Clients Table
Mobile_Number
Reg_IMEI
UserID
Mobile_Number
One_Time_Password
Server_nonce
Server_nonce2
Mobile_nonce
Mobile_Result
Client_Table
Server
Cryptography Layer
Client Certificate
Private Key
Installed
Decrypt QR code contents
Split QR code contents
S_UserID
URL
ns
Generate Mobile Nonce (n_mobile)
Extract IMEI
C_UserID, n_mobile, mn, IMEI
URL
lookup(mn) in "Registered-Clients" Table
Verify IMEI
Save (n_mobile) in Mobile-nonce
Generate a new server nonce (ns2) & save in Server_nonce2
S_UserID, n_mobile, ns2
Verify S_UserID & Mobile Nonce (n_mobile)
Verify S_UserID against received S_UserID
Append ns with (n_mobile)
ns
n_mobile
C_UserID, ns2, hash1
hash1
Hash
lookup (C_UserID) in "Client_Table"
Save (hash1) in Mobile_Result
Verify the server nonce (ns2)
Append Server-nonce with Mobile_nonce
Server_nonce
Mobile_nonce
Hash
hash2
S_UserID, n_mobile, Auth_Result
Compare both hashes
hash1
hash2
Display Authentication Result on Screen

Figure 3.8: 3LS-Authentication cryptography and response generation Layers

3.6 Conclusion

Based on the proposed objectives and the different authentication methodologies presented in chapter 1 and chapter 2, respectively. A number of aspects were considered while designing an online user authentication system for e-commerce applications. One of the most important aspects was to design a secure and user-friendly authentication system that overcomes Man-In-the-Browser (MITB) and Denial-of-Service (DOS) attacks while satisfying e-commerce security requirements. Furthermore, securing the protocol is considered as a mandatory requirement during the design phase, for satisfying security properties and overcoming adversaries, which is discussed in the verification phase in chapter 5.

In spite of the security mechanisms within most currently deployed user authentication systems, MITB attacks still pose serious threats that affect web-browsers. Most MITB solutions use the same infected browser to be authenticated via employing out-of-band communication channels. But the adversary could simply wait for the client to become authenticated before taking over the transaction. Thus, the proposed approach have transitioned from the MITB-infected traditional authentication mechanism to a different and detached mechanism that uses the mobile phone as the primary authentication device. The mobile phone is used along with its visual channel and cellular network due to its indispensability. The motivation behind employing two separate networks goes beyond the ability of the MITB to attack two detached devices within two unconnected authentication channels, to be linked to the same client. Therefore, the MUAS design is based on using the browser as an initiation to the authentication process, only requesting the registered mobile number of the client to mitigate the Denial-of-Service (DOS) attack. Upon verifying the mobile number, an OTP is sent to the mobile device to ensure that the primary authentication device is in possession by the initiating client. Subsequently, the authentication mechanism is transitioned to the mobile phone via a captured QR code, generated by the authentication server. The employment of an OTP within the MUAS is different than the OTP used within Microsoft [207] and Google's two-step verification process (2FA) [208], which is employed for providing a stronger authentication mechanism that protects the client's account in case of a stolen password. This two-step verification process requires the client to submit a static password (step 1) and an OTP (step 2), which can be sent to the client via e-mail, SMS, phone call, or by using an authenticator app.

Based on the presence of two detached networks, the proposed MUAS is composed of **two protocols**, the **challenge- and response-generation protocols**. Both of which are built on top of the cryptographic SSL/TLS protocol, where participants exchange their SSL-certificates, and share secret session keys in order to secure all exchanged messages, thus ensuring data privacy and integrity.

In terms of designing a secure QR code challenge within the challenge generation protocol, the system exploits some of the security attributes used in securing the protocol to become the foundation of the challenge, thus ensuring the uniqueness of the challenge and reducing the cost of computations. In addition, public key encryption is applied to the QR code contents, for increasing the security level and protecting the transmitted information from intruders.
After using the visual channel to capture the QR code, these attributes become the foundation for generating the response as well. Moreover, the challenge is composed of other components as well, some of which are exploited to initiate a new and secure connection with the authentication server, for submitting the response within the response generation protocol. While others are exploited to verify the identity of the authentication server to ensure its legitimacy.

With regard to representing the client in the response generation protocol, a simple mobile phone application called "3LS-Authentication" has been developed. The "3LS-Authentication" is a user-friendly application that is based on sharing the same SSL-certificate with the client's browser. It is composed of three levels of security: activation, cryptography and response generation. The activation layer represents the first security layer that request the client's credentials in order for the application to operate. In case of a stolen mobile phone, the activation layer only authorises the legitimate owner of the phone to use the application. The cryptography layer represents the second security layer that mitigates the DOS attack through extracting and submitting the IMEI to the server, for verifying a registered client. It also decrypts the QR code and exploit its contents to establish a secure cryptographic connection with the server, and starts to exchange and verify cryptographic messages with the server to secure the protocol against adversaries. The response generation layer is the third and final security layer that is based on exploiting and linking the QR-recovered unique content, along with one of the security attributes, required for securing the response generation phase. Thus, generating a response that combines unique security attributes from both phases to be finally hashed, using a one-way hash function. A one-way hash function was chosen to be applied as it is computationally infeasible to derive the original message from the hash result, even if it was intercepted by an adversary.

4 Implementation

4.1 Introduction

The previous design chapter proposed the theory of dividing the MUAS into two physically detached and logically connected phases. Moreover, it elucidated the two aspects of the client side within the challenge- and response-generation phases through browser and mobile phone connection with the server. It also illustrated how the server side is responsible for generating the challenge and verifying the response.

This chapter translates the proposed MUAS design principles and outlines the basic algorithms with regard to implementing the proposed authentication system. Based on the proposed MUAS split, each phase comprises several components that will be clarified in this chapter, together with descriptive flowcharts. Moreover, both phases utilise the same SSL-server as the framework platform for the challenge-generation and response-verification processes.

Taking advantage of the massive power, flexibility and performance that the C programming language offers [209], the SSL-server side and the SSL-client side (browser-based) are implemented with Embarcadero C++ Builder XE3 [210]. The client side (mobile phone-based) within the response-generation phase is implemented with Objective-C, using Xcode 6.1.1. Furthermore, a large section of the implementation functions was performed using software components and libraries provided by Chilkat Software, Inc. [211]. In addition, the QR code image-generation process is implemented using the ‘TBarCode Software Development Kit’ [212], which is a set of professional tools that supports all development environments and generates high-quality barcodes as printouts or graphic files. TBarCode SDK supports more than 100 different barcode symbologies and provides C/C++ developers with a barcode library (DLL), which is a full software library for embedding barcodes into applications and generating barcodes with only a few function calls [213].

This chapter demonstrates the implementation details and algorithms of the challenge- and response-generation phases separately, based on the grounds that the phases are not physically connected. The chapter is organised into the following sections: Section 4.2 provides an overview of the platform implementation that demonstrates the processes performed to establish an SSL/TLS connection between the server platform and both aspects of the client side within both phases. Section 4.3 presents the main algorithm of the challenge-generation phase. Section 4.4 illustrates the components of the challenge-generation phase required for forming the cryptographic QR code contents. Section 4.5 presents the response-generation phase algorithm. Section 4.6 illustrates the components of the response-generation phase, which are distributed between the mobile phone and the established SSL/TLS server platform. Section 4.7 concludes the implementation chapter.

4.2 Platform Implementation Overview

The prototype platform represents the foundation for both challenge- and response-generation phases. It consists of three main components, as presented in Figure 4.1. The three components are the challenge generator, the application and the verifier components.

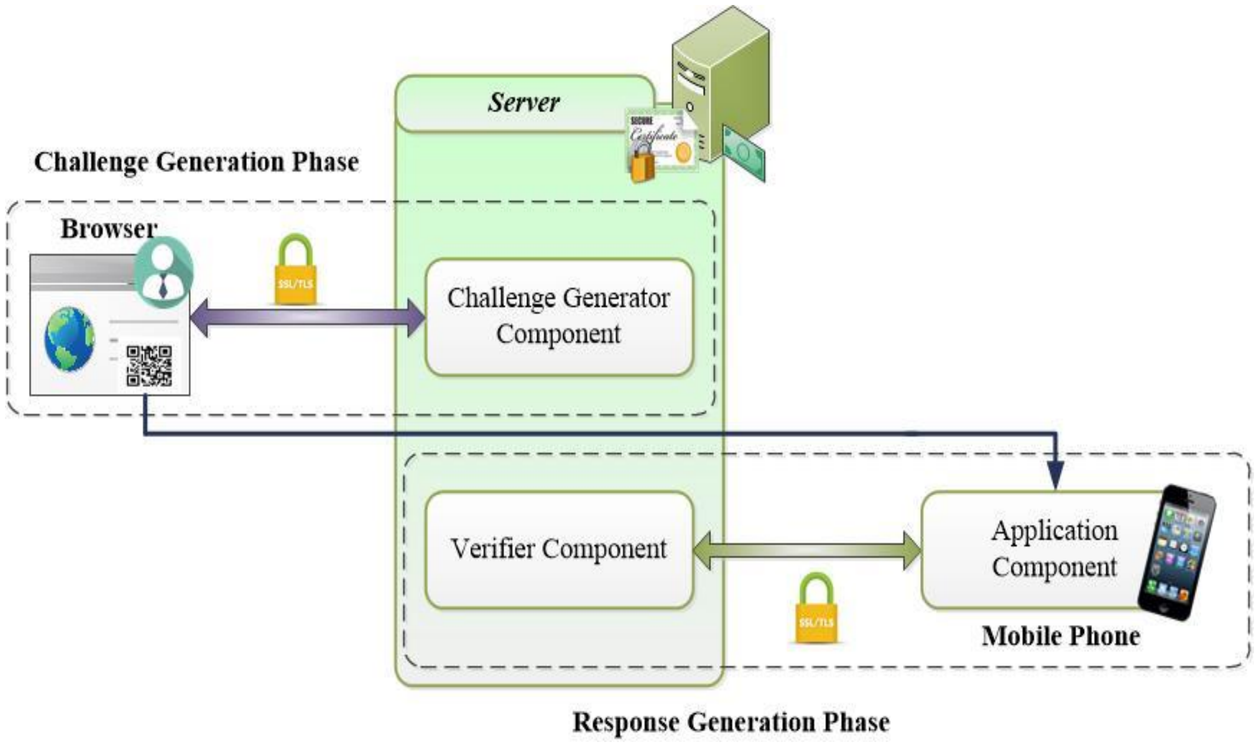

Server
Challenge Generation Phase
Browser
SSL/TLS
Challenge Generator Component
Verifier Component
Application Component
SSL/TLS
Mobile Phone
Response Generation Phase

Figure 4.1: Platform Implementation Overview

The first component residing within the server-side is the challenge-generator component, whereas the application and verifier components reside within the mobile phone and the server-sides, respectively.

Section 4.2.1 presents the general process of establishing the SSL/TLS server platform as the main prerequisite for the MUAS framework. In addition, the SSL/TLS client-side certificate is used to connect to the server, regardless of whether the client side is a browser or a mobile phone.

4.2.1Establishing an SSL/TLS Connection

The server platform of the proposed authentication system represents the required grounds for the logical connection to the physically separated challenge- and response-generation phases. However, the MUAS server platform is established to perform two fundamental tasks that represent the essence of the authentication system, namely challenge-generation and response-verification. Moreover, an SSL/TLS client certificate represents the client side of the challenge-generation phase. Whenever mutual authentication is required, the certificate is exchanged with the established SSL/TLS server side. In the challenge-generation phase, mutual authentication is mandatory for securing the generated response, whereas the response-generation phase does not require the SSL client certificate for the mobile phone. Figure 4.2 shows the SSL/TLS establishment process between the server platform and the client side.

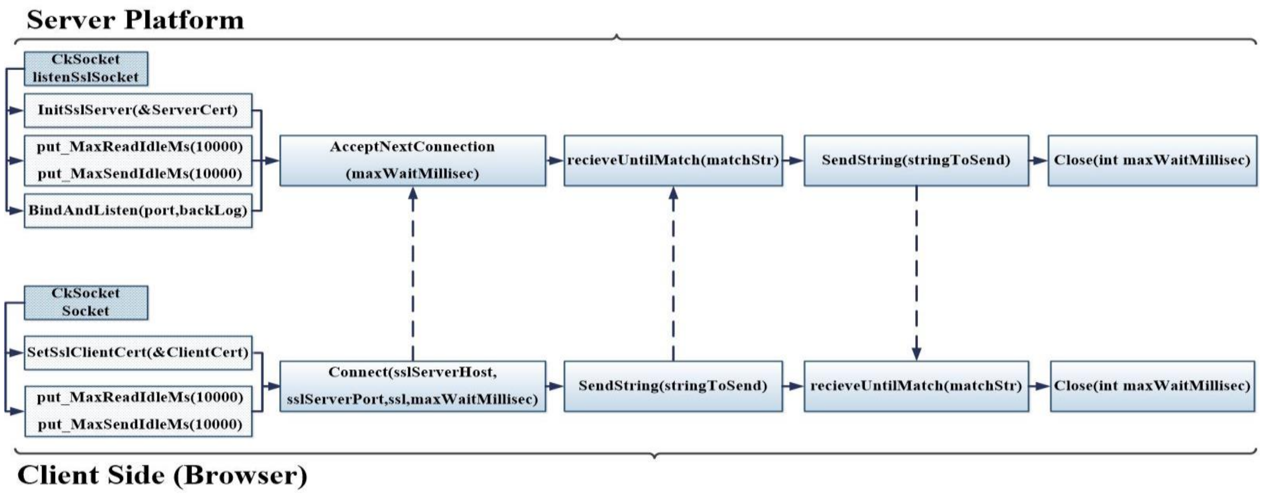

Server Platform
CkSocket listenSslSocket
InitSslServer(&ServerCert)
put_MaxReadIdleMs(10000)
put_MaxSendIdleMs(10000)
BindAndListen(port,backLog)
AcceptNextConnection (maxWaitMillisec)
recieveUntilMatch(matchStr)
SendString(stringToSend)
Close(int maxWaitMillisec)
CkSocket Socket
SetSslClientCert(&ClientCert)
put_MaxReadIdleMs(10000)
put_MaxSendIdleMs(10000)
Connect(sslServerHost, sslServerPort,ssl,maxWaitMillisec)
SendString(stringToSend)
recieveUntilMatch(matchStr)
Close(int maxWaitMillisec)
Client Side (Browser)

Figure 4.2: SSL/TLS Socket Establishment in a Client-Server Model

As illustrated in Figure 4.2, the server platform in both phases creates an SSL/TLS socket for accepting connections. The process begins by creating and initialising a TCP socket component with SSL/TLS capabilities for listening on a port to accept client connections [214]. Prior to accepting connections, the server application is initialised with the SSL/TLS server certificate to be used for SSL/TLS connections [214-216]. The TCP socket is then bound to a specified port to be configured for listening to incoming connections [214, 216].

Similarly, the same process applies on the client side to enable connection to the SSL/TLS server [214]. As mutual authentication is a prerequisite within the challenge-generation phase, the client is authenticated on the server through specifying the certificate via a certificate object [217]. This certificate is exploited later to secure the generated challenge.

The process of establishing an SSL/TLS handshake protocol between the client and the server requires several back-and-forth messages, negotiating the encryption algorithms and cryptographic keys to eventually set up a secure channel. The sending and receiving of these messages are controlled by the $MaxReadIdleMs$ and $MaxSendIdleMs$ properties, which are set in millisecond timescales to wait and to read/write. These steps are considered requirements to accept the SSL/TLS connection, as the handshake establishment protocol is part of the connection process [216, 217]. Thus, these properties must be set before accepting an incoming connection [214, 216]. Next, a secure SSL/TLS connection is established with the $remotehost{:}port$ using the $connect$ function, along with setting the SSL server hostname and port number [214, 217].

Once the connection is established, the server receives a confirmation message from the client. This is followed by the server confirming its connection. Subsequently, participants exchange their messages to generate the challenging QR code and to verify the received response within the challenge- and response-generation phases, respectively. Finally, the connection is closed after a predefined amount of time [214, 216].

4.3 Challenge-generation Phase

The challenge-generation phase signifies the launching phase of the proposed MUAS. It takes place between the client (browser) and the server platform within a secure SSL/TLS communication channel to generate a QR code to be displayed on the web browser. The server side is the focus of attention in this section, due to its responsibility for generating the challenge.

The server maintains a database in which a table called $Client - Table$ holds specific information related to each client-server session, including $UserID$, as the primary key of the table, as well as $Mobile\_Number$, $One\_Time\_Password$, $Server\_nonce$, $Server\_nonce2$, $Mobile\_nonce$ and $Mobile\_Result$.

Following the established mutual authentication between the client and the authentication server over the SSL/TLS communication channel, described in Section 4.2.1, a new record is created in the $Client - Table$, where the distinguished name (DN) within the received SSL-client certificate is saved in the primary key field $UserID$.

The sequence of steps leading to the generation of the challenging QR code is based on three messages exchanged between the authentication server and the browser. These messages are:

Msg1: *C_UserID, nb, mn,*

Msg2: *S_UserID, nb, ns,*

Msg3: *C_UserID, ns, OTP*

The authentication server depends heavily on the first and last messages, which includes the client's mobile number $\boldsymbol{mn}$ and one-time password $\boldsymbol{OTP}$, as they comprise the information required for executing the challenge-generation process. Whereas the fresh browser-nonce $\boldsymbol{nb}$, submitted in the first message and received again within the second message prevents an adversary from impersonating the server. These messages were composed based on the requirement of satisfying secrecy and authentication security properties, which will be discussed in Chapter 5.

However, the challenge generation process begins with verifying the received mobile number $\boldsymbol{mn}$ from $\boldsymbol{Msg1}$ against the primary key of the $\boldsymbol{Registered - Clients\,Table}$. Upon verification, a new record is created in the $\boldsymbol{Client - Table}$ for the C_UserID client, where the received identity C_UserID is saved in the $UsedID$ primary key field, and the received mobile number mn is saved in the $Mobile\_Number$ field of the same record. In ***Msg2***, the server generates a fresh nonce ns, which is saved in the $Server\_nonce$ field for verification purposes in both phases, as well as for security purposes.

For ensuring that the primary authentication device is in possession by the legitimate client, the server generates an OTP, saves it in the One_Time_Password field in the same record, and sends it to the mobile phone of the client via SMS. Upon reception, the client submits the OTP via browser. Next, upon receiving the ns value together with the submitted OTP in ***Msg3***, the server uses the attached C_UserID from *Msg3* to verify the received ns value against the $Server - nonce$ field for client verification. In the event of a match, the same attached C_UserID from *Msg3* is used to verify the received OTP value against the $One\_Time\_Password$ field. Based on the verification result, the server executes the final stage of the challenge-generation phase. In this stage, the server assembles, composes and encrypts the QR code contents using the client's public key that was extracted from the client certificate, received during the establishment of the SSL/TLS session. Finally, the QR code image is generated and displayed on the web browser of the initiating client. Figures 4.3 provide the overall algorithm of the challenge-generation phase. (Pseudo code is provided in Appendix A).

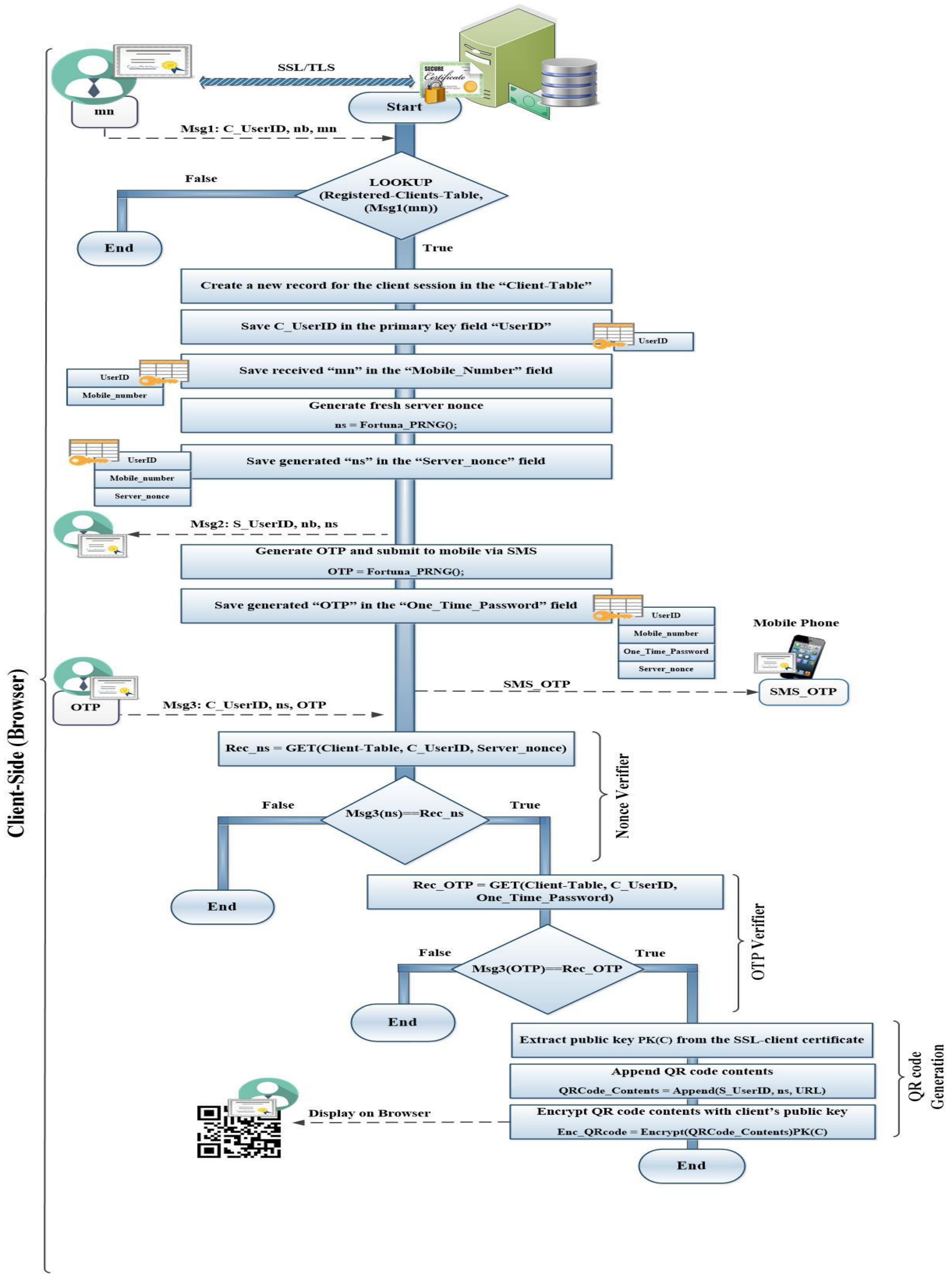

SSL/TLS
mn
Start
Msg1: C_UserID, nb, mn
False
LOOKUP (Registered-Clients-Table, (Msg1(mn))
End
True
Create a new record for the client session in the "Client-Table"
Save C_UserID in the primary key field "UserID"
UserID
Save received "mn" in the "Mobile_Number" field
UserID
Mobile_number
Generate fresh server nonce
ns = Fortuna_PRNG();
UserID
Mobile_number
Server_nonce
Save generated "ns" in the "Server_nonce" field
Msg2: S_UserID, nb, ns
Generate OTP and submit to mobile via SMS
OTP = Fortuna_PRNG();
Save generated "OTP" in the "One_Time_Password" field
UserID
Mobile_number
One_Time_Password
Server_nonce
Mobile Phone
SMS_OTP
SMS_OTP
OTP
Msg3: C_UserID, ns, OTP
Rec_ns = GET(Client-Table, C_UserID, Server_nonce)
False
Msg3(ns)==Rec_ns
True
Nonce Verifier
End
Rec_OTP = GET(Client-Table, C_UserID, One_Time_Password)
OTP Verifier
False
Msg3(OTP)==Rec_OTP
True
End
Extract public key PK(C) from the SSL-client certificate
Append QR code contents
QRCode_Contents = Append(S_UserID, ns, URL)
Display on Browser
Encrypt QR code contents with client's public key
Enc_QRcode = Encrypt(QRCode_Contents)PK(C)
QR code Generation
End
Client-Side (Browser)

Figure 4.3: Overall Challenge-generation Algorithm

4.4 Challenge-generation Components

The challenge-generation phase consists of three main components, which are responsible for generating a secure QR code challenge. The components are the **nonce component**, the **OTP component** and the QR code component. Figure 4.4 shows the three components, together with their procedures.

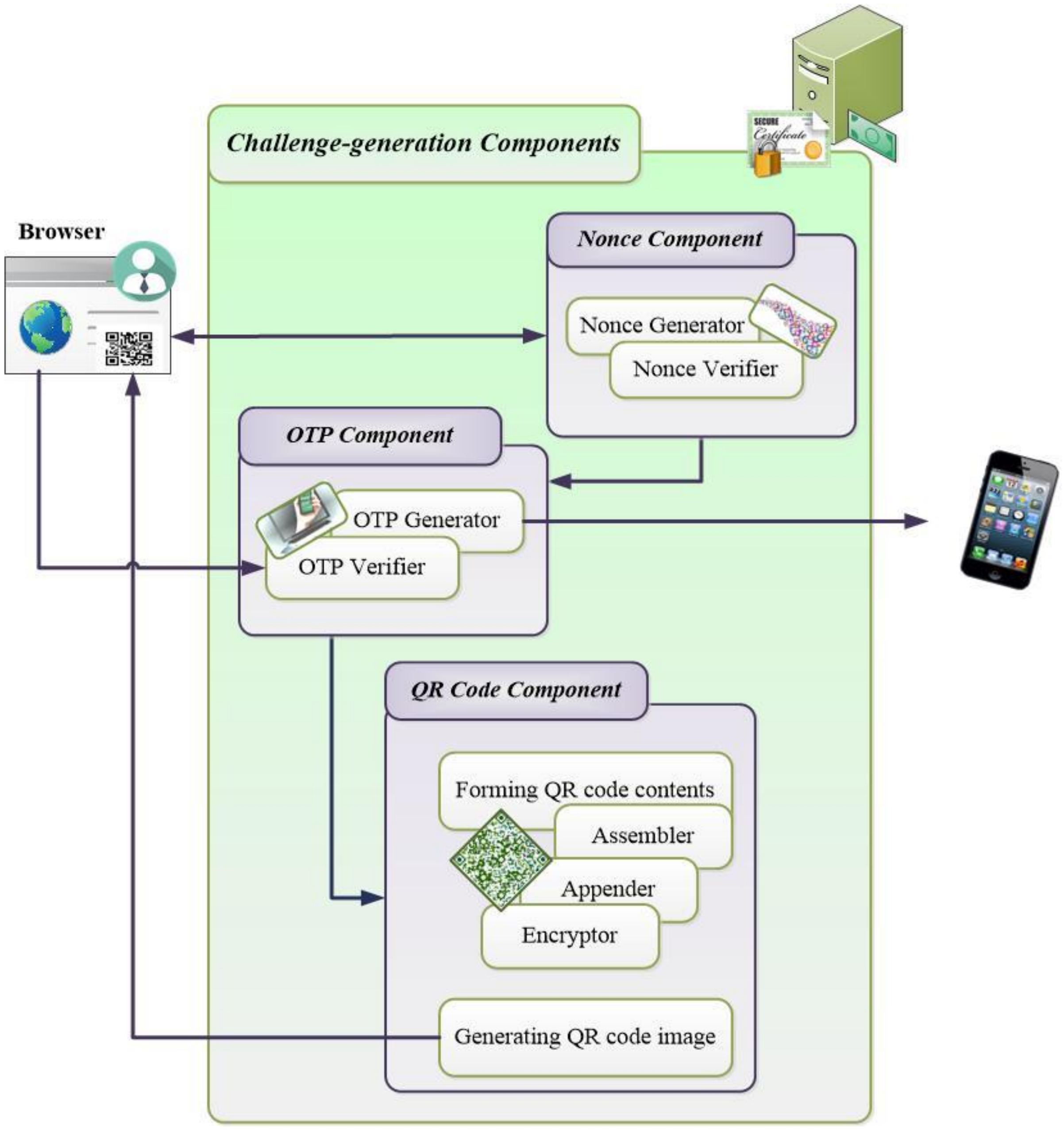


Figure 4.4: Challenge-generation Components

The three challenge-generation components can be summarized as follows:

- The **nonce component** is responsible for generating fresh random numbers. These numbers are used to secure messages exchanged between participants, in which verification is performed upon exchange. Moreover, this component is considered an essential feature in forming the QR code contents, since the freshly generated server nonce represents one of the contents that constitute the QR code.
- The **OTP component** generates a random number to be sent to the mobile phone via SMS. The client submits the received number through the browser for the server to verify upon reception. An OTP verifies that the mobile phone holder is a legitimate client who initiated the authentication system on the browser side.
- The **QR code component** is responsible for forming the contents of the QR code and for generating the QR code image.

The following sections present the challenge-generation component algorithms with regard to their sequence of events.

4.4.1Nonce Component

As part of developing a secure protocol, fresh values should be generated for each instantiation of each role [41]. Thus, each participant in the challenge-generation phase generates a fresh nonce to be exchanged and verified during communication.

The nonce component consists of two primary procedures that play an integral part in the challenge and response development mechanism. These procedures are executed in a sequential manner on behalf of the server platform whenever a client initiates a connection with the server through the established SSL/TLS channel. The procedures are the following:

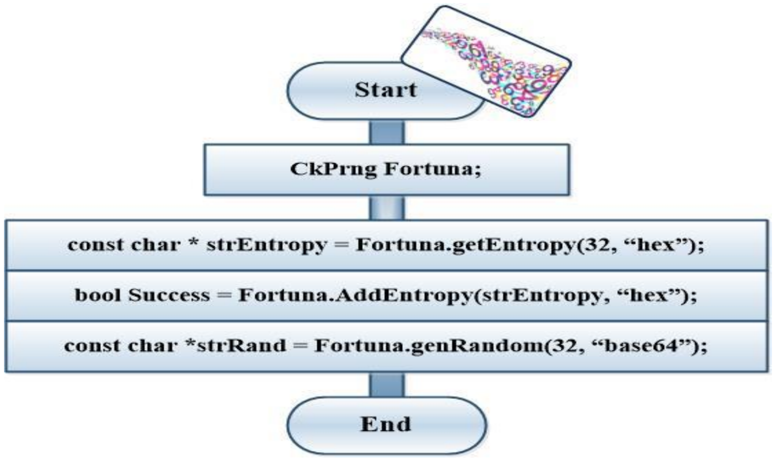
1. Nonce generator.
2. Nonce verifier.

The nonce generator and verifier procedures are mandatory in every secure protocol to prevent impersonators and malicious users from trying to replay the same message multiple times [41].

4.4.1.1 Nonce Generator

The generation of a fresh nonce within both phases is accomplished by employing the Fortuna Pseudo Random Number Generator (PRNG). The generator takes real random data (entropy) as a seed to generate an arbitrary amount of pseudorandom data. The accumulator then collects and pools the real random data from various sources to reseed the generator, which can be a difficult and time-consuming process in terms of implementation. Therefore, the pseudorandom data are used. Furthermore, new entropy data are added periodically in the event that further random data are needed [218].

The process of generating a pseudorandom number within MUAS begins with creating and initialising an object from the PRNG component [219]. The component implements the Fortuna PRNG algorithm using a 256-bit AES and SHA256, along with providing methods for accessing the system entropy. Next, the pseudorandom data generator reads entropy data from a system entropy source to generate an encoded string using the $GetEntropy$ function, which uses the $CryptGenRandom$ method within the Microsoft Crypto API to generate the pseudorandom number. The initial seeding of the generator should not exceed 32 bytes of entropy, as it should be specified within the function. The $AddEntropy$ function is used to seed the entropy into the generator to finally generate and return a specified number of random bytes in an encoded form using the $genRandom$ function [219, 220]. All generated nonce within both phases are specified as 32 bytes. Figure 4.5 provides the algorithm of the nonce generation procedure, along with the nonce exchange protocol and its subsequent procedures.



```
const char * strEntropy = Fortuna.getEntropy(32, “hex”);
bool Success = Fortuna.AddEntropy(strEntropy, “hex”);
const char *strRand = Fortuna.genRandom(32, “base64”);
```

Figure 4.5: Fresh Nonce Generation Procedure

The Fortuna PRNG function is used to generate random strings that represent the server nonce ns, the browser nonce nb and the mobile phone nonce mn. Furthermore, it is also employed within the OTP component to generate the OTP on the server side. However, the generated OTP is specified as 2 bytes.

Moreover, the nonce verifier procedure takes place upon nonce exchange to verify the received nonce against the original value that was generated and sent by the initiating party.

4.4.2QR Code Component

With regard to challenge generation, the QR code component represents the essence of the entire authentication system, which is considered the primary factor upon which the response-generation phase depends. It is composed of two primary procedures:

1. Forming QR code contents.
2. Generating QR code image.

The following sections illustrate the exact steps followed to generate the challenging QR code with regard to their sequence of events.

4.4.2.1 Forming QR Code Contents

Forming the QR code contents is a multi-level procedure that revolves around assembling the required components to form the QR code contents. The assembled contents are appended and encrypted as an extra layer of security against intruders and to ensure the authenticity of the client on both browser and mobile phone sides.

The following sections present and explain the algorithms of the three-level procedure that constitutes the QR code contents.

4.4.2.1.1 Assembling QR Code Contents

Assembling QR code contents is the first level of the formation procedure. It gathers the components that constitute the contents of the QR code. The components are as follows:

- The server's distinguished name S_UserID is the first component to be exported into the QR code contents. This component is mandatory for verifying proper server connectivity in the response-generation phase. The process works by importing the SSL-server certificate from the server platform to obtain the certificate's subject and full distinguished name (DN). As demonstrated in Figure 4.7, the process begins by creating an instance of a certificate store object [221], which represents a collection of certificates that can be loaded from a Windows-based certificate store or from a PFX file (PKCS# 12). Once the PFX file is loaded, the required certificate within the PFX is searched for by common name, subject, email address or by issuer and serial number [216].

- The second component is the freshly generated server nonce ns, which is considered the essence of the assembled QR code contents. This component is the foundation for the client's verification process within the response-generation phase. The nonce generation algorithm has been explained in Section 4.4.1.1

- The response end-point URL (URL) is the last component of the QR code content. This component represents the entry point to the intermediate cryptography layer. The address is formatted as '$remotehost$:$port$', which specifies the exact address of the server for connecting and sending the response for verification.

4.4.2.1.2 Appending QR Code Contents

Appending QR code contents is the second level of the formation procedure. It concatenates the server's distinguished name S_UserID, the server nonce ns and the response end-point URL. The appending process begins with creating an instance of a string class to be used with the $Append$ function [222]. When concatenating the contents, they are separated by a delimiter character to be used later for string splitting.

4.4.2.1.3 Encrypting QR Code Contents

Encrypting QR code contents is the final level of the formation procedure. The foundation of this procedure lies within the mutual authentication between the browser and the authentication server during the initial SSL/TLS channel establishment. The server exploits the received SSL-client certificate to secure the generated challenge.

For implementation, the received SSL-encoded string certificate is used to initialise a client certificate object $ClientCert$ [215]. The next step is to export the public key from the certificate object [215, 223] to obtain the public key in XML format [224]. Finally, the appended QR code contents are encrypted by creating an instance of the '$CkRsa$' object [225] to be used for setting the encoding mode and importing the extracted public key [225, 226], which is used to perform the encryption process. Figure 4.6 shows the algorithm of the overall QR code contents formation process.

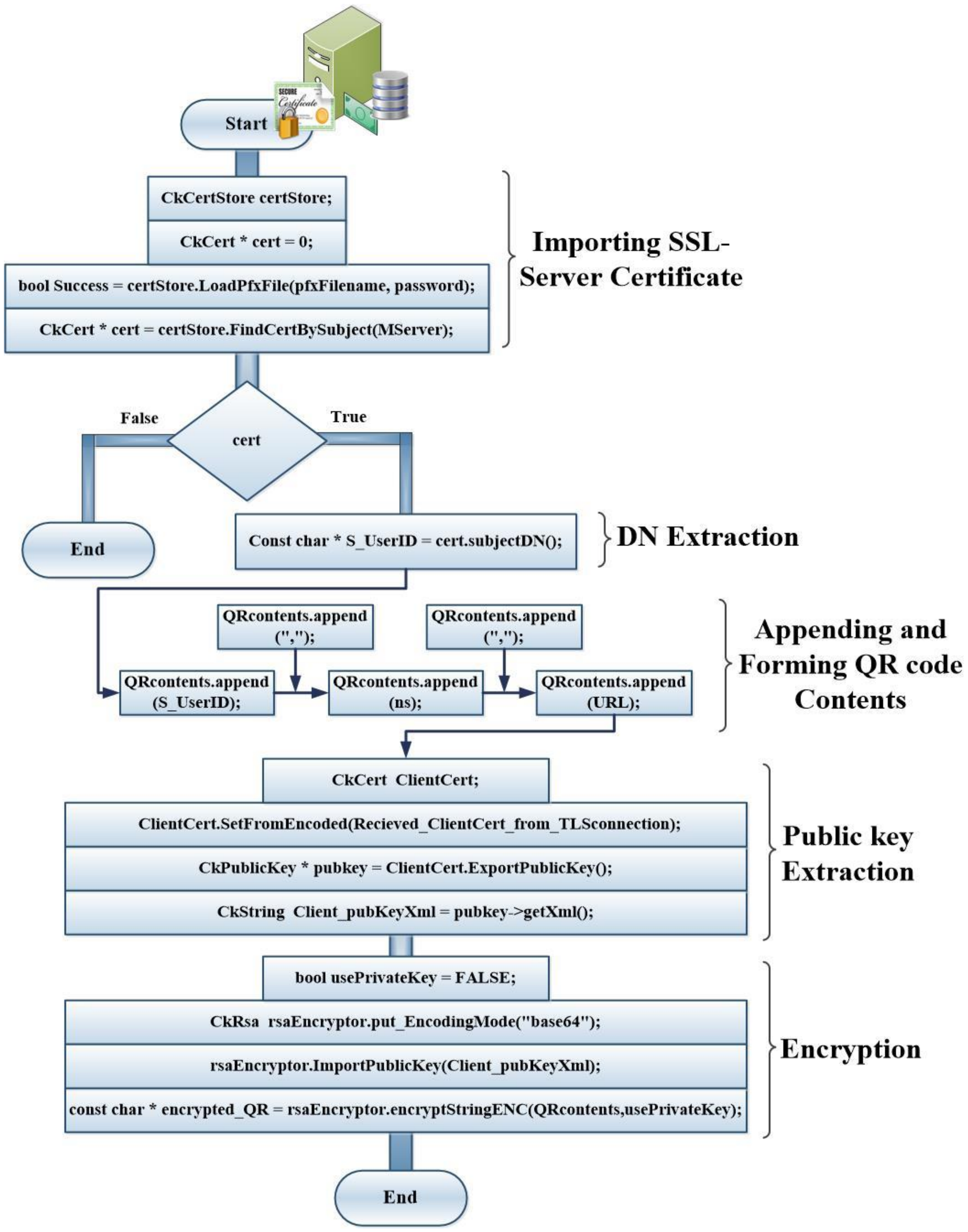

Start
CkCertStore certStore;
CkCert * cert = 0;
bool Success = certStore.LoadPfxFile(pfxFilename, password);
CkCert * cert = certStore.FindCertBySubject(MServer);
Importing SSL-Server Certificate
False
True
cert
End
Const char * S_UserID = cert.subjectDN();
DN Extraction
QRcontents.append (",");
QRcontents.append (",");
QRcontents.append (S_UserID);
QRcontents.append (ns);
QRcontents.append (URL);
Appending and Forming QR code Contents
CkCert ClientCert;
ClientCert.SetFromEncoded(Recieved_ClientCert_from_TLSconnection);
CkPublicKey * pubkey = ClientCert.ExportPublicKey();
CkString Client_pubKeyXml = pubkey->getXml();
Public key Extraction
bool usePrivateKey = FALSE;
CkRsa rsaEncryptor.put_EncodingMode("base64");
rsaEncryptor.ImportPublicKey(Client_pubKeyXml);
const char * encrypted_QR = rsaEncryptor.encryptStringENC(QRcontents,usePrivateKey);
Encryption
End

Figure 4.6: Forming QR Code Contents

4.4.2.2 Generating the QR Code Image

Following the formation of the QR code contents, the second procedure begins and generates the QR code image to be displayed on the browser side. The QR code is then captured by the client's mobile phone camera to be used for client authentication and verification.

The QR code image-generation procedure is performed by allocating memory for the barcode structure. The next step is to specify and set up some of the QR code attributes, such as the contents to be encoded and the QR code size. The size, in device pixels, is determined by performing barcode height and width calculations [227]. Figure 4.7 shows the process of generating a barcode on behalf of the server side to be displayed on the client side (browser).

The generation procedure begins with calling the function $BCAlloc$ to set up and initialise the barcode's internal structure by allocating memory and retrieving a barcode handle pBC. The required barcode type, and the text to be encoded, are then specified through the $BCSetBCType$ and $BCSetTextA$ functions, respectively. The barcode pattern is then prepared to be drawn or saved through creating an internal barcode representation from the previous parameters set by calling the function $BCCreate$ [227]. With regard to the output device resolution, performing calculations in device pixels specifies the barcode size. The calculations involve converting the height and width (given in millimetres) to inches, where the result is multiplied by the resolution in dots per inch (dpi) of the output device. Next, functions $BCGet2D\_XRows$ and $BCGet2D\_XCols$ return the number of modules calculated for the chosen barcode, thus returning the horizontal and vertical module count. The height and width are optimised by rounding up their values via the *Ceil* function, to be multiplied next by the module count. Finally, the optimised height and width can be used to draw the barcode or to save it as an image file by calling the function $BCSaveImage$ [227, 228].

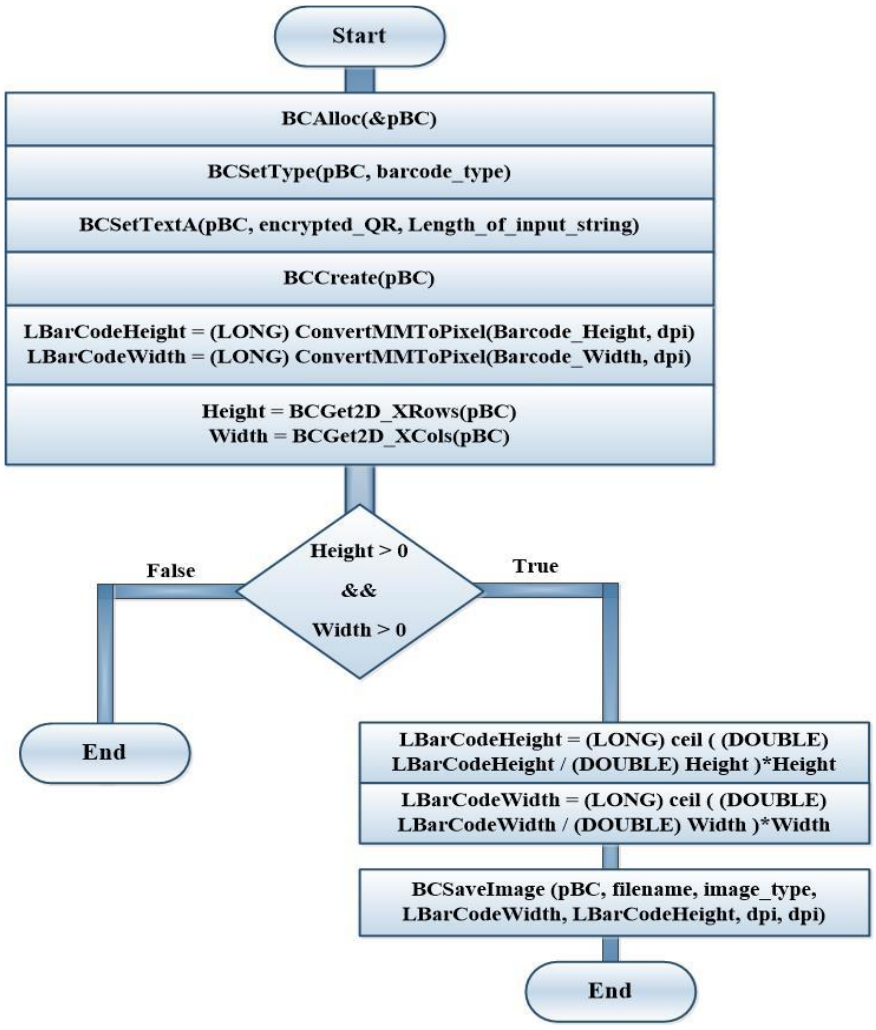

Start
BCAlloc(&pBC)
BCSetType(pBC, barcode_type)
BCSetTextA(pBC, encrypted_QR, Length_of_input_string)
BCCreate(pBC)
LBarCodeHeight = (LONG) ConvertMMToPixel(Barcode_Height, dpi)
LBarCodeWidth = (LONG) ConvertMMToPixel(Barcode_Width, dpi)
Height = BCGet2D_XRows(pBC)
Width = BCGet2D_XCols(pBC)
Height > 0
&&
Width > 0
False
True
End
LBarCodeHeight = (LONG) ceil ((DOUBLE) LBarCodeHeight / (DOUBLE) Height)*Height
LBarCodeWidth = (LONG) ceil ((DOUBLE) LBarCodeWidth / (DOUBLE) Width)*Width
BCSaveImage (pBC, filename, image_type, LBarCodeWidth, LBarCodeHeight, dpi, dpi)
End

Figure 4.7: Generating the QR Code Image Process

4.5 Response-generation Phase

The response-generation phase is the second and final phase of the proposed MUAS. The process is based on developing a single-view mobile phone application, called ‘3LS-Authentication’, which communicates with the authentication server after capturing the displayed QR code for verification purposes. The response-generation phase is composed of three layers, namely an activation layer, a cryptography layer, and a response-generation layer. The activation and cryptography layers take place on the mobile phone side within the ‘3LS-Authentication’ application, whereas the response-generation layer takes place between the application and the server side.

The sequence of steps leading to generation of the response within the ‘3LS-Authentication’ begins

with the activation layer, where the client is requested to submit the credentials that was set during the app installation on the mobile phone. This layer works as a defence mechanism against stolen mobiles, where the application credentials are required from the legitimate client in order to operate. The second layer extracts the IMEI from the mobile phone, for the server to verify against the registered IMEI. It also executes the decryption and splitting processes to recover the QR code contents. These are utilised for several purposes in the second and third layers.

- The first content to be utilised within this phase is the server’s response end-point URL, which is responsible for establishing an SSL/TLS connection with the authentication server, as shown in the cryptography layer in Figure 4.8. This paves the way for sending the generated response.

- The second recovered content to be exploited is the distinguished name DN of the server, which ensures the authenticity of the connected server by verifying it against the DN of the SSL-server certificate, and against the received server’s identity, as illustrated in the response-generation layer in Figure 4.8.

- The final recovered content is the fundamental element of the response-generation mechanism, namely the fresh server nonce. This nonce is utilised for the logical connection of the challenge- and response-generation phases. In addition, it plays an essential part in generating the response, as illustrated in the response-generation layer in Figure 4.8.

Finally, the third layer passes on the hashed result to the server for verification, along with the exchanged nonce from the response-generation phase. Figure 4.9 shows the detailed steps of the ‘3LS-Authentication’ application within the response-generation phase. (Pseudo code is provided in Appendix A).

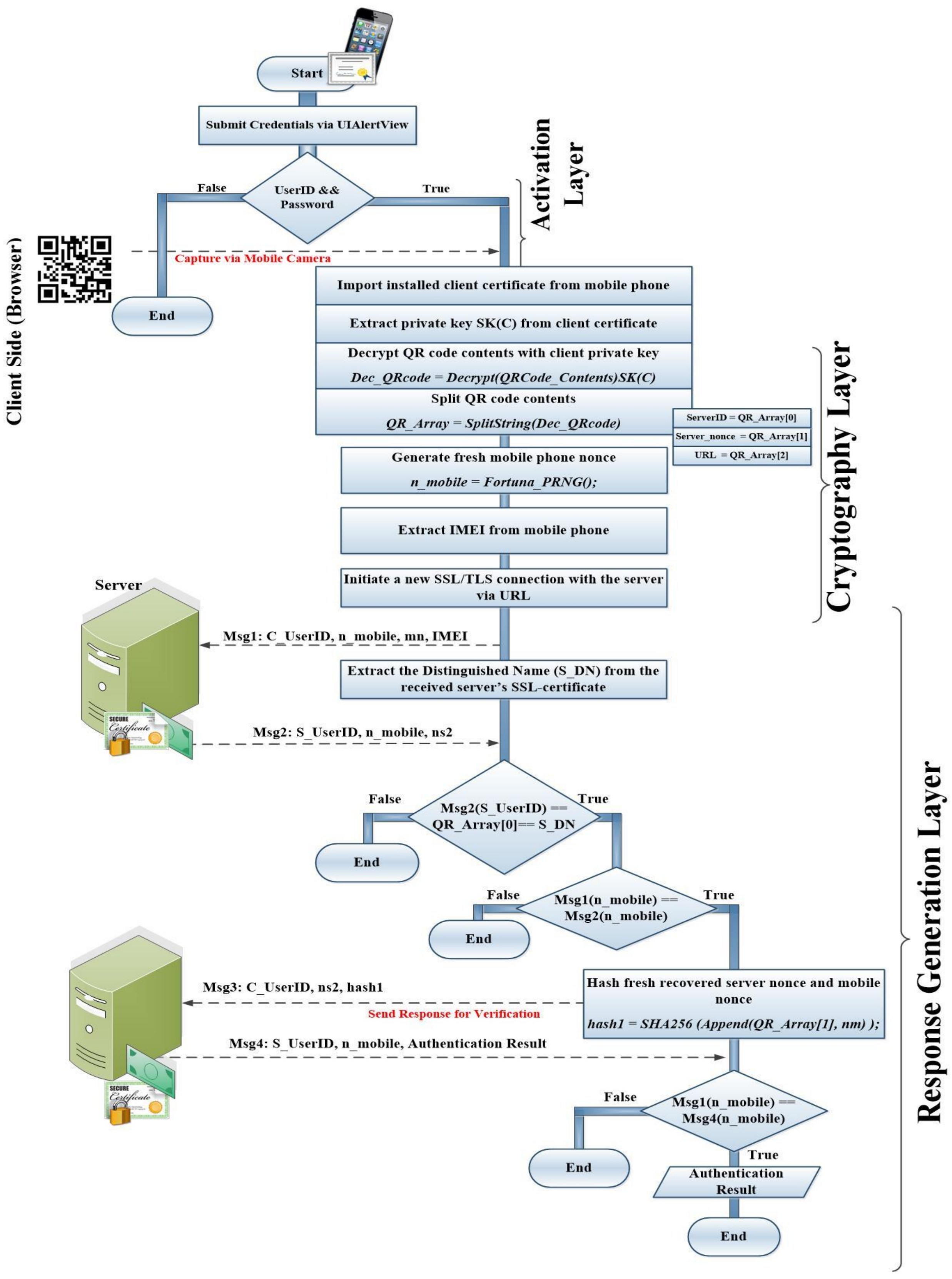

Start
Submit Credentials via UIAlertView
UserID && Password
False
True
Activation Layer
Client Side (Browser)
Capture via Mobile Camera
End
Import installed client certificate from mobile phone
Extract private key SK(C) from client certificate
Decrypt QR code contents with client private key
Dec_QRcode = Decrypt(QRCode_Contents)SK(C)
Split QR code contents
QR_Array = SplitString(Dec_QRcode)
ServerID = QR_Array[0]
Server_nonce = QR_Array[1]
URL = QR_Array[2]
Cryptography Layer
Generate fresh mobile phone nonce
n_mobile = Fortuna_PRNG();
Extract IMEI from mobile phone
Initiate a new SSL/TLS connection with the server via URL
Server
Msg1: C_UserID, n_mobile, mn, IMEI
Extract the Distinguished Name (S_DN) from the received server's SSL-certificate
Msg2: S_UserID, n_mobile, ns2
False
True
Msg2(S_UserID) == QR_Array[0]== S_DN
End
False
True
Msg1(n_mobile) == Msg2(n_mobile)
End
Response Generation Layer
Msg3: C_UserID, ns2, hash1
Send Response for Verification
Hash fresh recovered server nonce and mobile nonce
hash1 = SHA256 (Append(QR_Array[1], nm));
Msg4: S_UserID, n_mobile, Authentication Result
False
Msg1(n_mobile) == Msg4(n_mobile)
True
End
Authentication Result
End

Figure 4.8: Mobile Phone-side Process of the Response-generation Phase

4.6 Response-generation Components

The response-generation phase consists of two main components, namely an application component and a verifier component. These components are enclosed within the mobile phone and server-side end-point for performing response generation and response verification, respectively. Both of these comprise a number of procedures for performing specific actions.

The application component residing on the mobile phone side consists of several procedures for securing the application against stolen phones, together with exploiting the recovered QR code contents for response generation. The recovered QR code contents are also utilised to establish a secure SSL/TLS communication channel for submitting the response to the server for verification. Furthermore, the application component is responsible for securing the protocol from impersonators via a nonce exchanging mechanism. The verifier component residing on the server-side is responsible for verifying the received response. Figure 4.9 shows the main components within the response-generation phase.

Moreover, a common component exists between both parties, namely the nonce component, which is responsible for generating and verifying exchanged nonces among participants. This component was explained previously in Section 4.4.1.

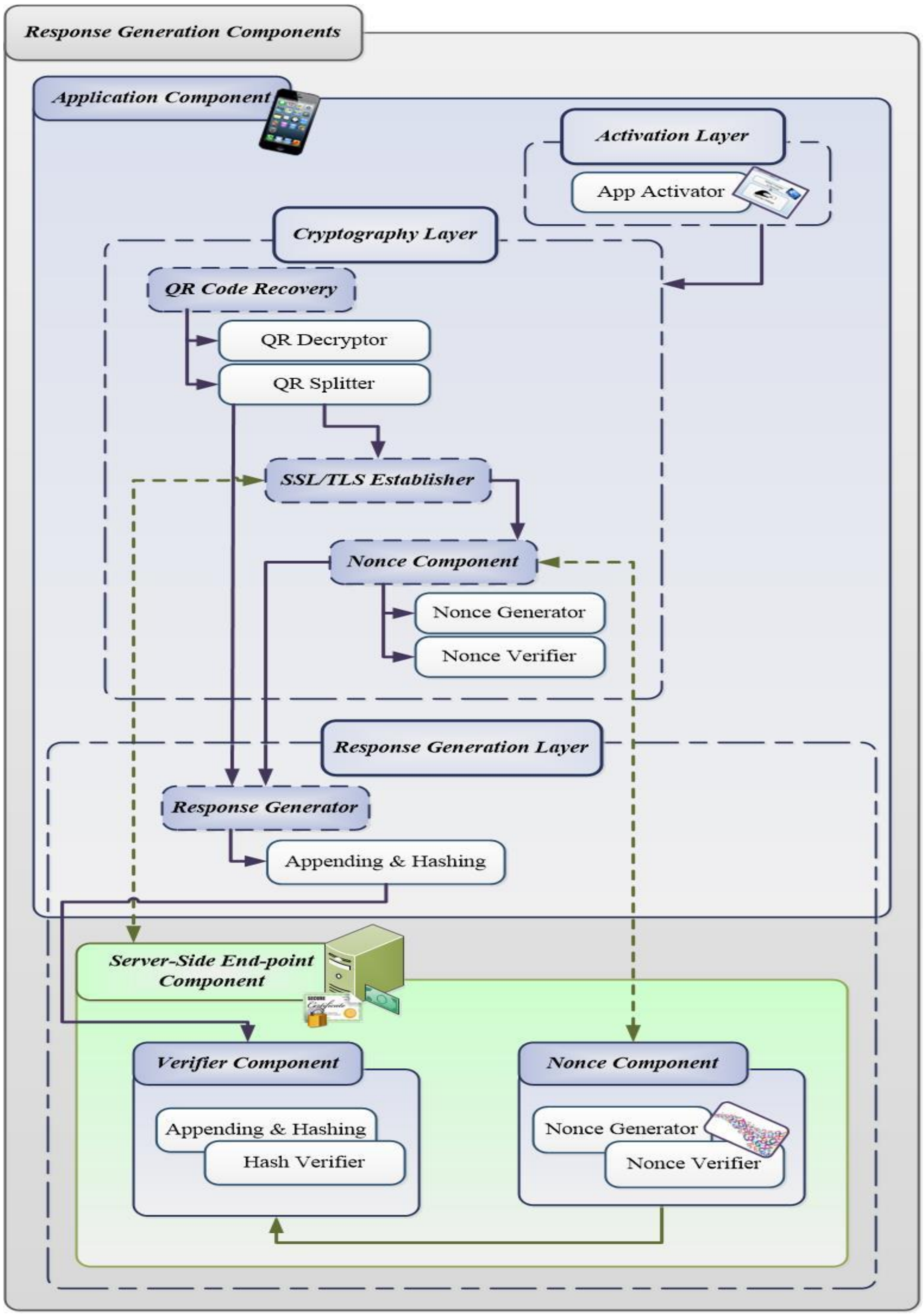


Figure 4.9: Response-generation Components

Based on the components constituting the response-generation phase, the sequence of steps leading to the generation of the response within the mobile phone side are described in Section 4.6.1, whereas the steps for verifying the response within the server side are described in Section 4.6.1.5.

4.6.1Application Component

The mobile phone application developed for authenticating on-line clients via the QR code comprises **three layers**. The first layer is the **activation layer**, which is responsible for activating the app through a single procedure, *the app activator*. The second layer is the **cryptography layer**, which is responsible for decrypting and recovering the QR code contents, in addition to establishing a secure cryptographic communication channel with the server. The third and final layer is the **response-generation layer** that is distributed between the mobile phone application and the authentication server end-point for response generation and response verification, respectively. In this section, the algorithms of the first and second layers are illustrated, along with the mobile phone's response-generation algorithm. The algorithm of the server's response-verification is illustrated in Section 4.6.1.5. The four primary procedures used within the cryptography layer and the response-generation algorithms are as follows:

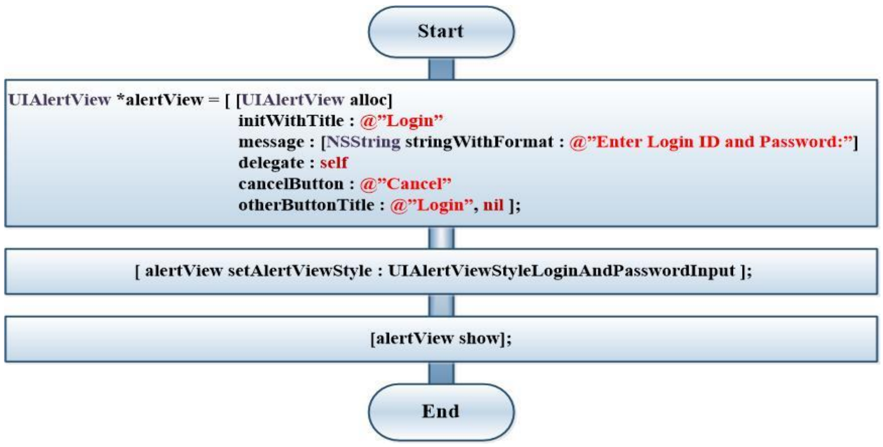
1. QR code recovery.
2. SSL/TLS establisher.
3. Nonce component.
4. Response generator.

These procedures are executed in a sequential manner on behalf of the mobile phone whenever the QR code is captured. The following sections present the algorithms for the procedures of the application component with regard to their sequence of events.

4.6.1.1 App Activator

The app activator procedure lies within the activation layer. It is performed on behalf of the mobile phone application to ensure the legitimacy of the user. When loading the application, the user is requested to submit his/her credentials via an alert view. Thus, a titled custom button ‘Login’ is added to the alert view.

The alert view is configured through its properties and methods to set the title, message, delegate and configure buttons. Following the creation and initialisation of the alert view, the presentation style of the alert is set to display two text fields that accommodate the login ID and password. Figure 4.10 shows the steps for creating and initialising the alert view, together with the style settings.



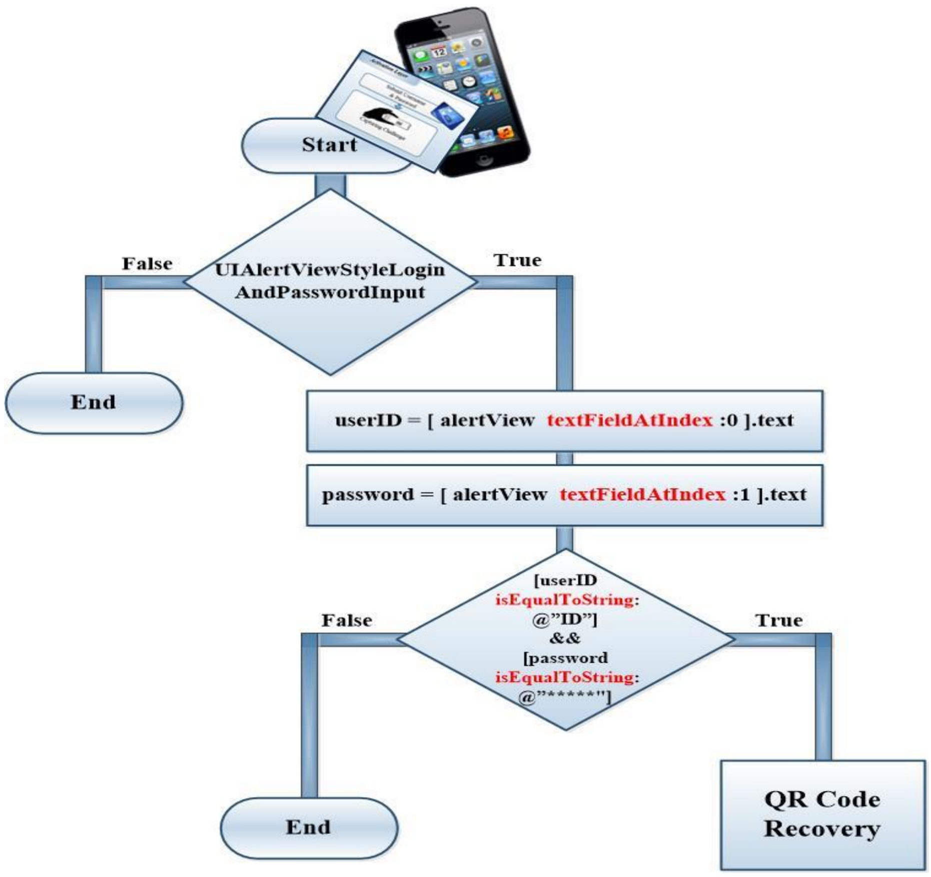
```
UIAlertView *alertView = [ [UIAlertView alloc]
                              initWithTitle : @”Login”
                              message : [NSString stringWithFormat : @”Enter Login ID and Password:”]
                              delegate : self
                              cancelButton : @”Cancel”
                              otherButtonTitle : @”Login”, nil ];

[ alertView setAlertViewStyle : UIAlertViewStyleLoginAndPasswordInput ];

[alertView show];
```

Figure 4.10: Alert View Creation and Settings

Due to the existence of a custom button within the alert, a delegate is designated to handle the button action. Whenever the custom button is tapped, the delegate responds with the $alertView : clickedButtonAtIndex$ method. Within the alert view, the $textFieldAtIndex$ method retrieves the values entered by the client from the text fields. The values are verified against the application credentials that were stored on the mobile phone during the installation process of the application. Figure 4.11 shows the $clickedButtonAtIndex$ response upon tapping the custom button. Finally, and after verifying the credentials’ authenticity, the alert view is discarded and the cryptography layer begins to operate.



Start
False
UIAlertViewStyleLogin AndPasswordInput
True
End
userID = [alertView textFieldAtIndex :0].text
password = [alertView textFieldAtIndex :1].text
[userID isEqualToString: @"ID"] && [password isEqualToString: @"******"]
False
True
End
QR Code Recovery

Figure 4.11: Alert View Behavior

4.6.1.2 QR Code Recovery

As the captured QR code contains encrypted data, the first component to handle the QR code contents is the QR code recovery component, which comprises two primary procedures:

1. QR code decryptor.
2. QR code splitter.

The procedures are responsible for decrypting and splitting the QR code contents to generate the response. The following sections present the algorithms of the QR code recovery procedures with regard to their sequence of events.

4.6.1.2.1 QR Code Decryptor

The foundation of the QR code decryptor lies within installing the client's digital certificate on the mobile phone; therefore, the client certificate is embedded within the application's main bundle. The embedded certificate is the same X.509 certificate of the PC, which is used for authenticating the browser within the challenge-generation phase. The QR code decryptor procedure fetches the private key from the installed certificate for decrypting the QR code contents.

From an implementation point of view, the process begins by creating a bundle object for the client to specify the required certificate [229]. This is achieved by selecting the required certificate's name and type. Followed by authenticating the certificate via its password in order to be used within the application. The selected certificate is then initialised into a static data object [230] for later reuse within the SSL/TLS communication channel establishment. Consequentially, the certificate is loaded and its private key is exported [231] to obtain the private key in XML format [232] by using a certificate object [231]. Finally, the exported private key decrypts the QR code contents by creating an instance of the '$CkRsa$' object to perform the decryption process [233]. Prior to content decryption, the encoding mode is set, and the private key is imported into the RSA object for performing the decryption process [234]. Figure 4.12 shows the process of extracting the private key from the client certificate, along with the decryption process.

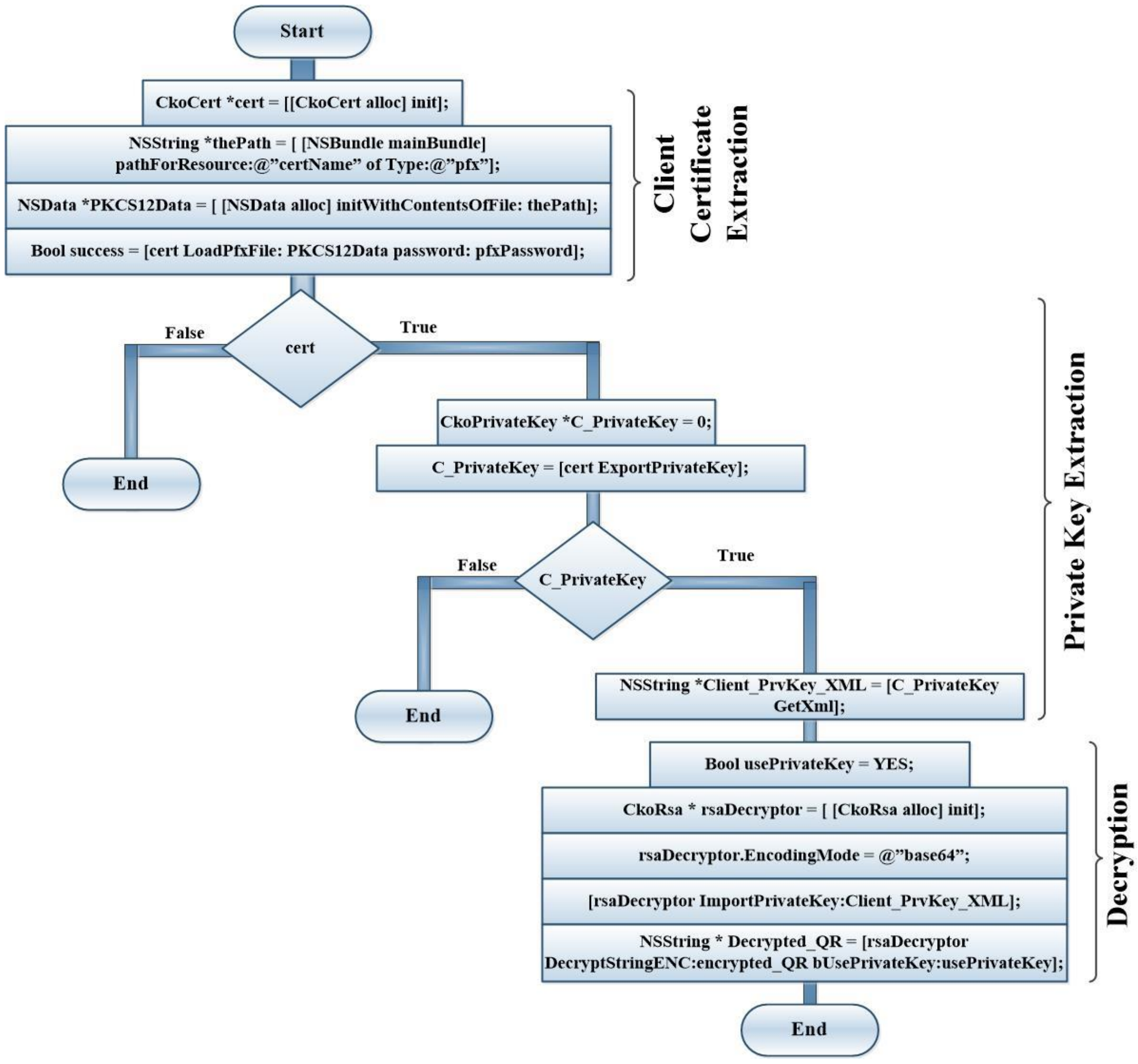

Start
CkoCert *cert = [[CkoCert alloc] init];
NSString *thePath = [[NSBundle mainBundle] pathForResource:@"certName" of Type:@"pfx"];
NSData *PKCS12Data = [[NSData alloc] initWithContentsOfFile: thePath];
Bool success = [cert LoadPfxFile: PKCS12Data password: pfxPassword];
Client Certificate Extraction
cert
False
True
End
CkoPrivateKey *C_PrivateKey = 0;
C_PrivateKey = [cert ExportPrivateKey];
C_PrivateKey
False
True
End
NSString *Client_PrvKey_XML = [C_PrivateKey GetXml];
Private Key Extraction
Bool usePrivateKey = YES;
CkoRsa * rsaDecryptor = [[CkoRsa alloc] init];
rsaDecryptor.EncodingMode = @"base64";
[rsaDecryptor ImportPrivateKey:Client_PrvKey_XML];
NSString * Decrypted_QR = [rsaDecryptor DecryptStringENC:encrypted_QR bUsePrivateKey:usePrivateKey];
Decryption
End

Figure 4.12: Extracting the Client Certificate and its Private Key to Perform Decryption

4.6.1.2.2 QR Code Splitter

Following the QR code decryption process, the contents are split into their original components using the delimiter character as a separator to return an array containing the QR code components [235]. The recovered components are the server's distinguished name DN, the fresh server nonce and the server's end-point (URL). Figure 4.13 shows the steps of splitting the decrypted QR code contents.

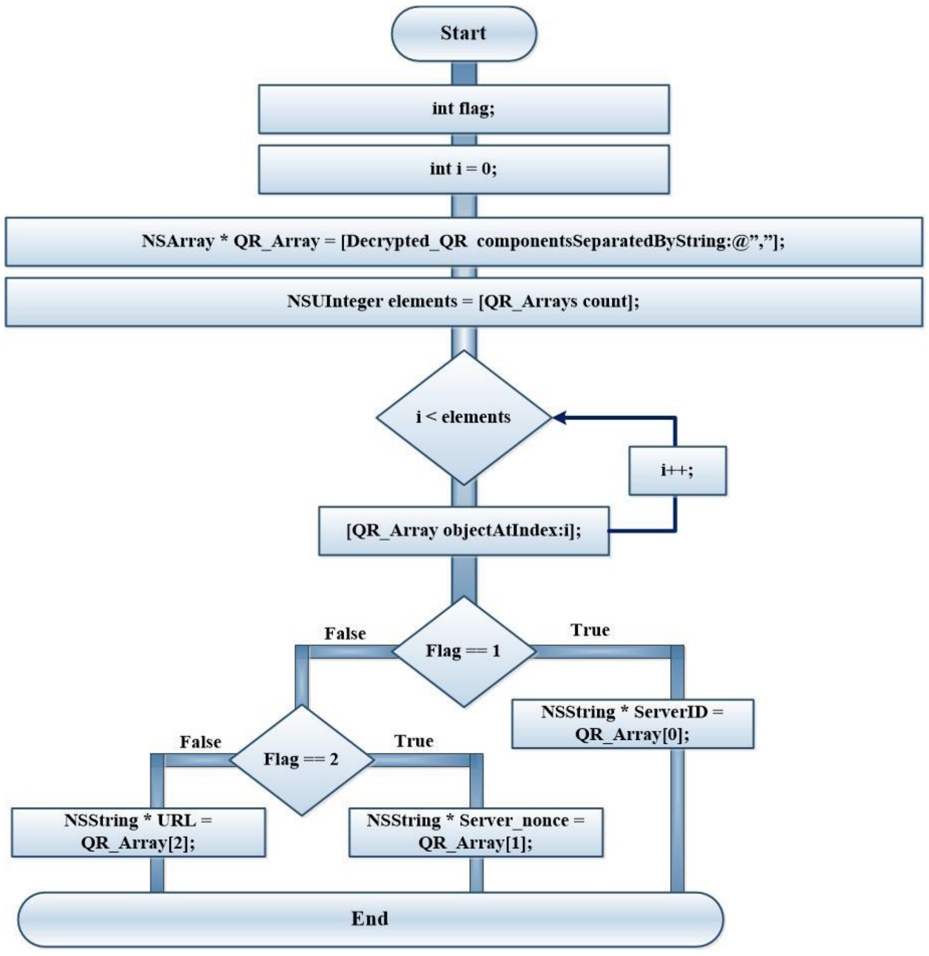

Start
int flag;
int i = 0;
NSArray * QR_Array = [Decrypted_QR componentsSeparatedByString:@”,”];
NSUInteger elements = [QR_Arrays count];
i < elements
i++;
[QR_Array objectAtIndex:i];
False
Flag == 1
True
NSString * ServerID = QR_Array[0];
False
Flag == 2
True
NSString * URL = QR_Array[2];
NSString * Server_nonce = QR_Array[1];
End

Figure 4.13: Splitting the QR Code Contents

4.6.1.3 SSL/TLS Establisher

Paving the way to send the generated response over a secure channel for verification, an SSL/TLS connection with the server should be established. The QR code-recovered URL ‘$remotehost$:$port$’, from the previous section represents the basis for the SSL/TLS establishment process. Thus, the address is split into the required fields for connecting to the SSL server. Figure 4.14 shows the steps for connecting to a specific SSL server [236].

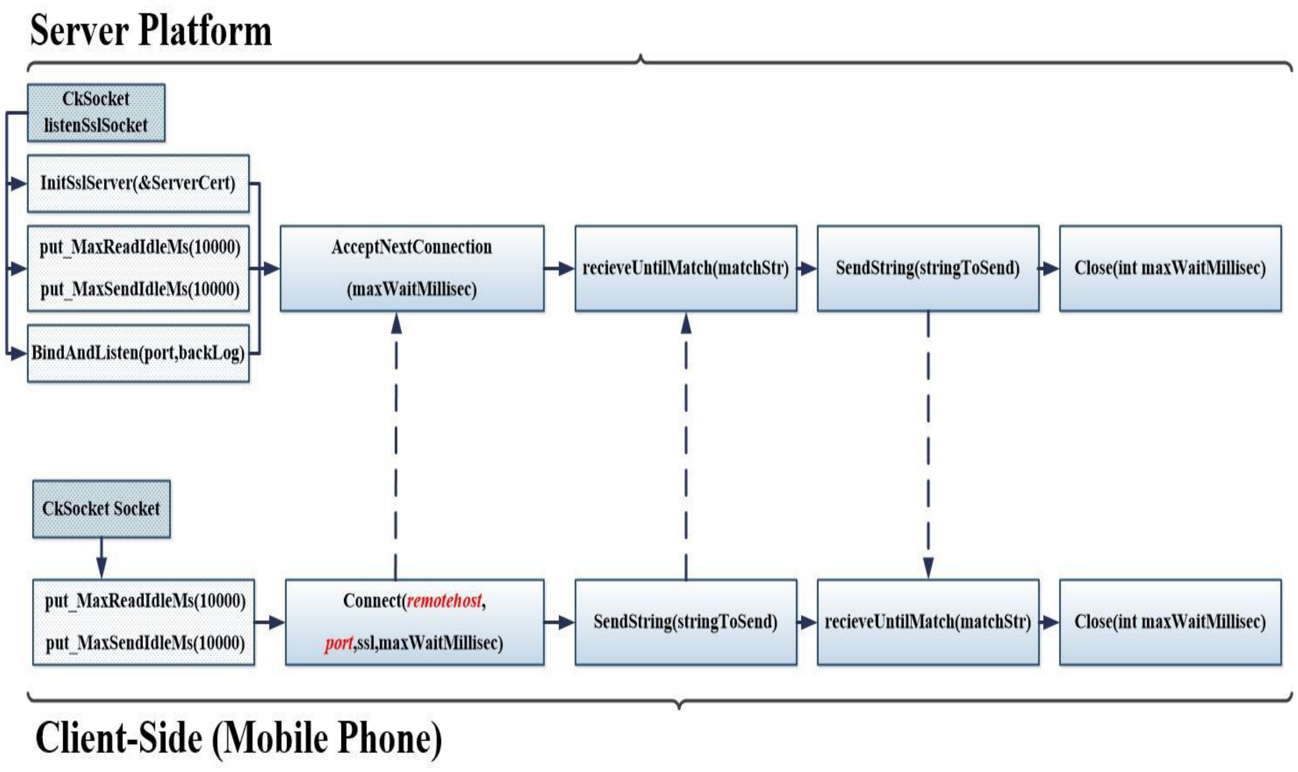

Server Platform
CkSocket listenSslSocket
InitSslServer(&ServerCert)
put_MaxReadIdleMs(10000)
put_MaxSendIdleMs(10000)
BindAndListen(port,backLog)
AcceptNextConnection (maxWaitMillisec)
recieveUntilMatch(matchStr)
SendString(stringToSend)
Close(int maxWaitMillisec)
CkSocket Socket
put_MaxReadIdleMs(10000)
put_MaxSendIdleMs(10000)
Connect(remotehost, port,ssl,maxWaitMillisec)
SendString(stringToSend)
recieveUntilMatch(matchStr)
Close(int maxWaitMillisec)
Client-Side (Mobile Phone)

Figure 4.14: Establishing SSL/TLS Client-side Process within the Response-generation Phase

Following the SSL/TLS channel establishment, both parties start exchanging their freshly generated nonce as a means to secure exchanged messages from impersonators and to satisfy security properties. As part of the current SSL/TLS handshake establishment process, the server submits his/her SSL certificate for authentication. Hence, the QR code-recovered server identity $ServerID$ from the previous section is verified against the distinguished name S_DN of the received SSL server certificate, and against the server identity S_UserID received from the server during the second message $Msg2$. This triple verification helps to verify the identity of the connected server, which prevents excess overhead on behalf of the mobile phone in the event of unverified identity.

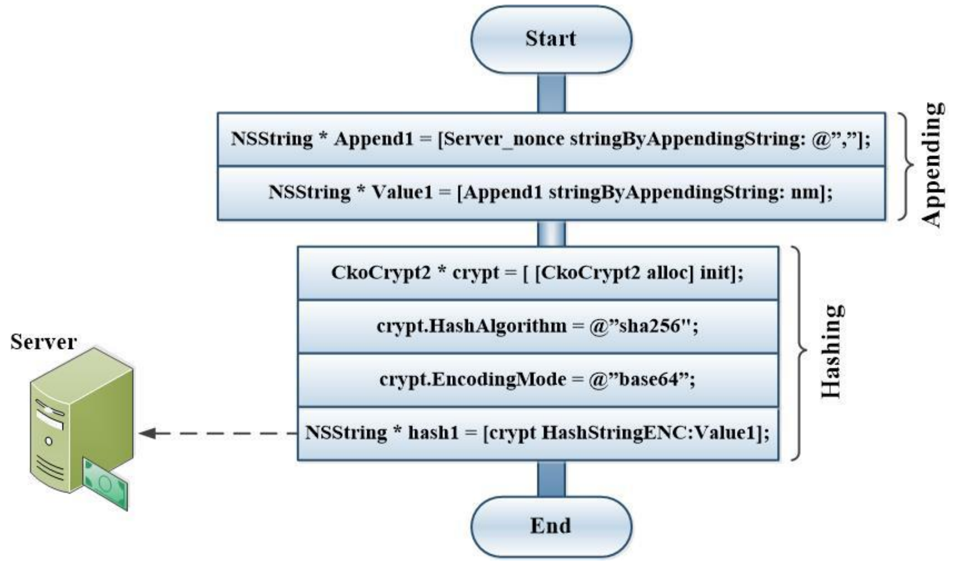
4.6.1.4 Response Generator

Following the process of securing the communication channel against impersonators and verifying the legitimacy of the connecting party, the response generator procedure begins to execute its own procedures, namely appending and hashing.

The procedure is responsible for generating the response from the QR code-recovered nonce along with the fresh mobile phone nonce. The following section presents the algorithm of the response-generator procedure with regard to the sequence of events. Figure 4.15 shows the hashing algorithm, along with the encryption process.

4.6.1.4.1 Appending and Hashing

The appending and hashing procedure is based on exploiting the recovered contents of the QR code, specifically the server nonce $Server\_nonce$, which is utilised to generate the response. Moreover, the procedure exploits the freshly generated nonce of the mobile phone nm, which is appended to the $Server\_nonce$. Following this, the hashing algorithm $(SHA256)$ is applied on the appended value, thus generating the response to be sent to the server for verification. The hashing process is based on creating and initialising an encryption object [237, 238], followed by setting its properties and performing the hashing process [238].



```
NSString * Append1 = [Server_nonce stringByAppendingString: @”,”];
NSString * Value1 = [Append1 stringByAppendingString: nm];
CkoCrypt2 * crypt = [ [CkoCrypt2 alloc] init];
crypt.HashAlgorithm = @”sha256";
crypt.EncodingMode = @”base64”;
NSString * hash1 = [crypt HashStringENC:Value1];
```

Figure 4.15: Response Hashing and Encryption Processes

Finally, and enclosed in *Msg3*, the hashed response is sent to the server over the established SSL/TLS channel, together with the mobile phone’s identity and exchanged nonce for verification.

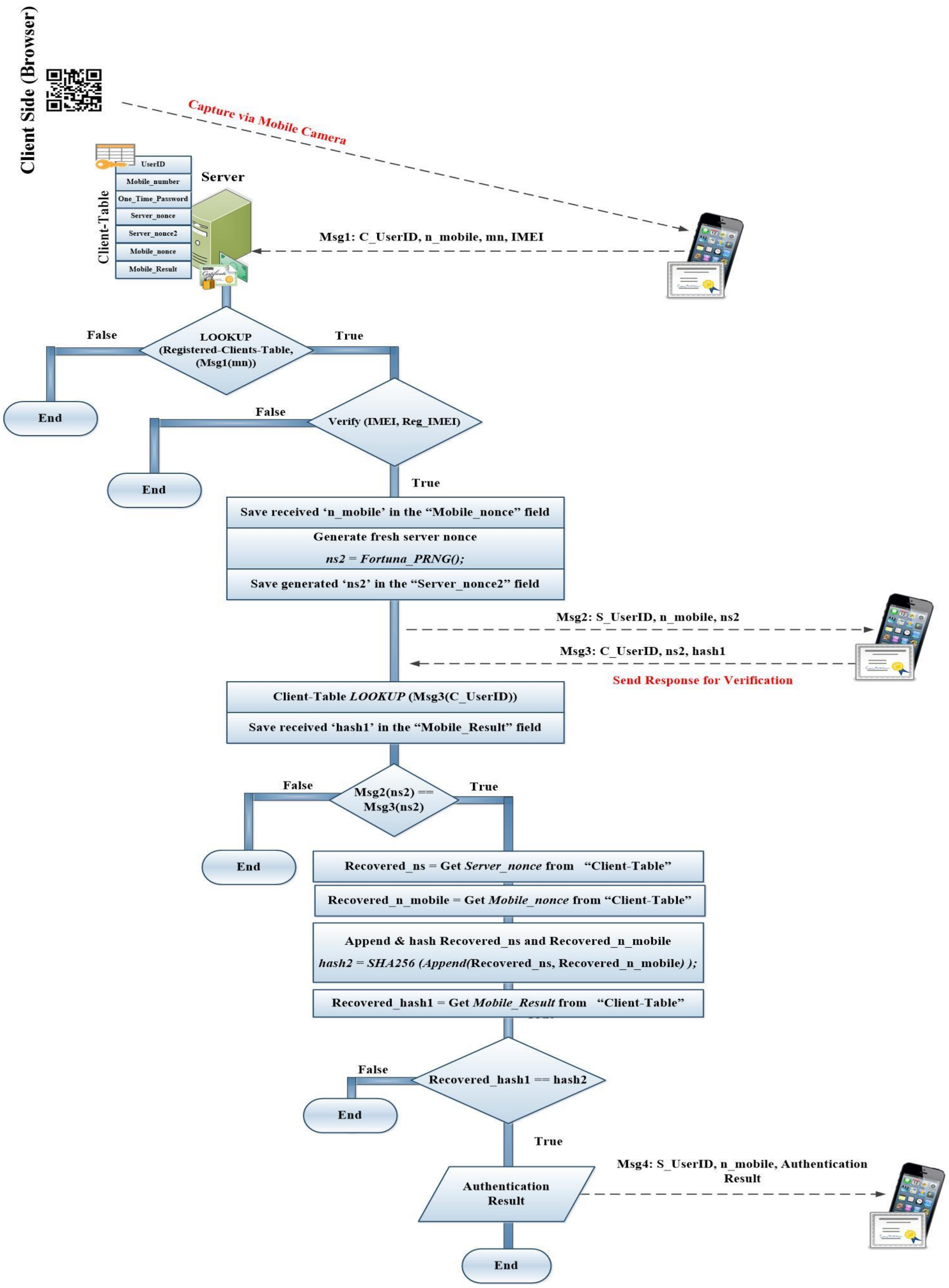
4.6.1.5 Verifier Component

The verifier component is the final component within the server side. It is responsible for validating the received response, and consequently for verifying the authenticity of the client. This component consists of two primary procedures, which are:

1. Appending and hashing.
2. Hash verifier.

The appending and hashing procedure works in exactly the same way as the appending and hashing in the response-generator component as shown in Figure 4.16. However, the exact and detailed steps of the verifier component are described in this section.

Upon receiving *Msg3*, the server side verifies the response as the final step of the authentication mechanism. The verification process starts with exploiting the received mobile phone's identity C_UserID against the $UserID$ field in the $Client - Table$ to store the received response $hash1$ in the $Mobile - Result$ field for later verification. Following this, the mobile phone nonce nm and the original server nonce ns that was generated previously in the challenge-generation phase are recovered from the $Mobile\_nonce$ and $Server\_nonce$ fields, respectively. Both of the recovered values are appended and hashed using the $SHA256$ hashing algorithm, thus forming $hash2$ to eventually be compared against $hash1$ to finally verify the authenticity of the client. Figure 4.16 shows the detailed steps in the final verification layer.



Client Side (Browser)
Capture via Mobile Camera
Client-Table
UserID
Mobile_number
One_Time_Password
Server_nonce
Server_nonce2
Mobile_nonce
Mobile_Result
Server
Msg1: C_UserID, n_mobile, mn, IMEI
False
LOOKUP (Registered-Clients-Table, (Msg1(mn))
True
End
False
Verify (IMEI, Reg_IMEI)
End
True
Save received 'n_mobile' in the "Mobile_nonce" field
Generate fresh server nonce
ns2 = Fortuna_PRNG();
Save generated 'ns2' in the "Server_nonce2" field
Msg2: S_UserID, n_mobile, ns2
Msg3: C_UserID, ns2, hash1
Send Response for Verification
Client-Table LOOKUP (Msg3(C_UserID))
Save received 'hash1' in the "Mobile_Result" field
False
Msg2(ns2) == Msg3(ns2)
True
End
Recovered_ns = Get Server_nonce from "Client-Table"
Recovered_n_mobile = Get Mobile_nonce from "Client-Table"
Append & hash Recovered_ns and Recovered_n_mobile
hash2 = SHA256 (Append(Recovered_ns, Recovered_n_mobile));
Recovered_hash1 = Get Mobile_Result from "Client-Table"
False
Recovered_hash1 == hash2
End
True
Msg4: S_UserID, n_mobile, Authentication Result
Authentication Result
End

Figure 4.16: Verification Algorithm of the Response-generation Phase

4.7 Conclusion

Building on the design model in the previous chapter, the aim of this chapter is to (1) implement the proposed system, (2) provide a demonstration of both MUAS challenge- and response-generation phases, (3) show all exchanged messages between participants within both phases, which will be verified in the next chapter, and (4) execute the proposed novelty within the implementation phase. Hence, this chapter paves the way for the evaluation chapter (chapter 5), in terms of performance measurement.

This chapter can be summarised as follows:

- Based on the design model described in Chapter 3, this chapter focuses on applying the proposed methodology to build the MUAS model/framework with regard to the SSL/TLS socket creation that accepts connections from both phases consecutively. The SSL/TLS socket is the foundation for the logical connection of the physically separated phases through the generated challenge, causing the response-generation phase to be dependent on the challenge-generation phase.
- Placing significant emphasis on the SSL/TLS sessions between participants contributes to securing both phases. In the challenge-generation phase, the SSL-client certificate is utilised to verify the identity of both participants and to secure the generated challenge. In the response-generation phase, the distinguished name extraction from the received SSL-server certificate helps to prevent mobile phone overheads as a result of verifying it against the received server identity and against the recovered server identity included within the QR code.
- The implementation of the response-generation phase depends on the outcome of the challenge-generation phase, along with the sharing of a common certificate for proceeding with the response-generation process.
- With regard to the provision of a secure and consistent protocol with a unique challenge, the server's freshly generated nonce is utilised within the challenge-generation phase to provide a dynamic nonce-based QR code. Furthermore, utilising the mobile phone's fresh nonce, to be combined with the QR code-recovered nonce in the response-generation phase, contributes to connecting both sessions logically, despite being physically separated. Thus, protocol security is achieved even if the server nonce is compromised.

5 Evaluation

5.1 Introduction

Research results in formal protocol verification area showed that formal verification tools have helped avoid standardising protocols with security defects [239]. Therefore, this chapter aims at reliably evaluating and verifying the proposed Mobile-User-Authentication System (MUAS) security protocols with respect to Man-in-the-Browser (MITB) attack by using the Scyther security protocol verification tool [240]. However, formal verification of a security protocol is performed with respect to a formal model [241]. A formal model comprises a protocol model that presents how the protocol should behave using static description [41, 241], an execution model that formalises dynamic aspects in terms of traces [41, 241], an adversary model that is derived from the protocol description and defined as a set of terms known to the compromised agents [41, 241], and a specification of the exact security properties that the protocol should satisfy [41, 241].

The organisation of this chapter is as follows. Section 5.2 provides an overview of both security protocols constituting the proposed MUAS. Section 5.3 conducts a security and performance evaluation through theoretical and implementation performance measurements, respectively. The theoretical evaluation provides a detailed description of the protocols, and justifies the corresponding messages within both protocols along with their contents with respect to possible attacks. However, the implementation evaluation is conducted in terms of computational and communication costs, with respect to the most related work. Section 5.4 introduces the adversary model of the proposed MUAS. Section 5.5 provides the mandatory protocol specification, for performing the required verification process and generating the results. Section 5.6 provides the verification results by applying the Scyther verification tool on both protocols, along with a representative analysis of the compromised protocol, and an in-depth analysis of the satisfied security properties. Section 5.7 concludes the findings of the verification chapter.

5.2 MUAS Protocol Overview

In a world of insecure networks, cryptographic protocols play a significant role when designing a secure authentication system by employing cryptography to achieve and satisfy security properties [41, 242]. Likewise, cryptographic input values (freshly-generated nonce) are considered indispensable for satisfying the security properties within secure protocols, as they play an essential role in cryptographic security, such as in stream ciphers, block ciphers, modes of authentications and message authentication codes [243]. Therefore, a combination of both mechanisms were cautiously employed within the proposed MUAS to provide a secure authentication protocol with respect to satisfying security properties [41].

The proposed approach circumvents the MITB capabilities through transitioning the authentication mechanism to a new and detached medium that uses the mobile phone of the client as the primary authentication device. Thus, a transaction can be carried out via the mobile phone without submitting confidential information through an infected browser as discussed in the future work in chapter 6. Therefore, the MUAS addresses the problem of the MITB attack via two protocols: the challenge-generation protocol and the response-generation protocol. The challenge-generation protocol starts with the untrusted computer establishing an SSL/TLS communication channel with the server, thus generating a shared secret key; security attributes exchanged between the untrusted computer and the server, required information submitted by the untrusted computer and a specific security attribute verified by the server to eventually generate the challenge, representing the basis behind the transition to the response-generation protocol through utilising the visual channel.

The response-generation protocol starts with activating the mobile phone application, capturing and decoding the challenge; establishing an SSL/TLS communication channel with the server, generating a shared secret key; security attributes exchanged between the mobile phone and the server; and response generated and sent to the server for verification. Figure 5.1 depicts the MUAS security protocol overview.

In terms of verification, there are three basic roles that constitute the system: browser $\boldsymbol{B}$, server $\boldsymbol{S}$, and the mobile phone $\boldsymbol{M}$. Although $\boldsymbol{B}$ and $\boldsymbol{M}$ represent different devices, they represent the same client. Thus, they share the same digital certificate on different platforms, denoted by $CERT_C$. Consequently, the public and secret key pair of the client on both platforms are represented as $pk(C)/sk(C)$, respectively.

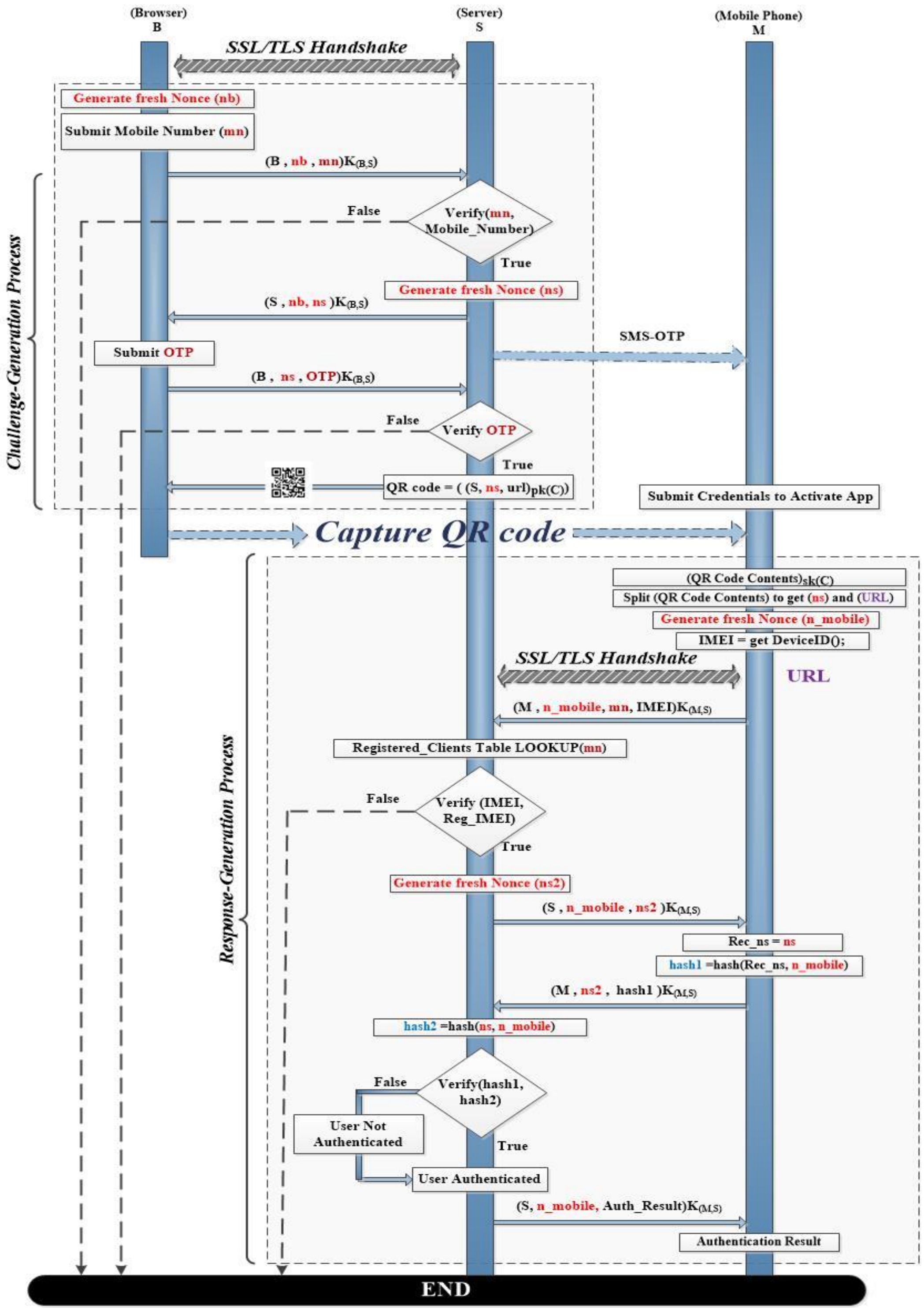

(Browser) B
(Server) S
(Mobile Phone) M
SSL/TLS Handshake
Generate fresh Nonce (nb)
Submit Mobile Number (mn)
(B , nb , mn)K(B,S)
False
Verify(mn, Mobile_Number)
True
Generate fresh Nonce (ns)
(S , nb, ns)K(B,S)
SMS-OTP
Submit OTP
(B , ns , OTP)K(B,S)
False
Verify OTP
True
QR code = ((S, ns, url)pk(C))
Submit Credentials to Activate App
Capture QR code
Challenge-Generation Process
(QR Code Contents)sk(C)
Split (QR Code Contents) to get (ns) and (URL)
Generate fresh Nonce (n_mobile)
IMEI = get DeviceID();
SSL/TLS Handshake
URL
(M , n_mobile, mn, IMEI)K(M,S)
Registered_Clients Table LOOKUP(mn)
False
Verify (IMEI, Reg_IMEI)
True
Generate fresh Nonce (ns2)
(S , n_mobile , ns2)K(M,S)
Rec_ns = ns
hash1 =hash(Rec_ns, n_mobile)
(M , ns2 , hash1)K(M,S)
hash2 =hash(ns, n_mobile)
False
Verify(hash1, hash2)
User Not Authenticated
True
User Authenticated
(S, n_mobile, Auth_Result)K(M,S)
Authentication Result
Response-Generation Process
END

Figure 5.1: MUAS Security Protocol Overview

5.3 MUAS Security and Performance Evaluation

Chapters 3 and 4 presented the theoretical design and implementation outlines of the proposed protocol, respectively. This section conducts and presents the results of evaluating the proposed protocol from a theoretical and implementation point of view.

With respect to possible attacks, the theoretical performance measurement provides a justification of the exchanged messages and their contents, along with the attached security attributes and the employed cryptographic mechanisms. Whereas the implemented performance measurement evaluates the proposed protocol by conducting a comparison against the most related work in terms of computational and communication costs. In addition to measuring the execution time of the protocol.

5.3.1Theoretical Performance Measurement

The challenge- and response-generation protocols are built on the basis of establishing SSL/TLS communication channels between participants with mutual and server SSL-certificate authentication, respectively, where the authenticity of the SSL-certificates is verified via a certificate authority (CA). Consequentially, both protocols build up to establish 128-bit secret session keys $\boldsymbol{k}(\boldsymbol{B},\boldsymbol{S})$ and $\boldsymbol{k}(\boldsymbol{M},\boldsymbol{S})$ for AES encryption. A 128-bit key size is sufficient enough to symmetrically encrypt and secure exchanged messages within both protocols against adversaries and impersonators [244]. The AES algorithm is considered the most popular symmetric cipher [157], due to its ability to encrypt an entire block of 128-bits using key sizes of 128, 192 or 256-bits [157]. Unlike other symmetric ciphers, such as DES and 3DES, which are slow and operate on small blocks with a single key size [157]. However, in terms of encryption performance, speed, secure memory storage, flexibility, and efficiency in hardware and software, AES surpasses both of them [165].

With regard to generating a secure protocol that satisfies the security properties, all exchanged messages comprise an identity of the message originator as a proof-of-identity and proof-of-origin. The identity is 64 bytes in size, which is constructed from the Distinguished Name (DN) field values of the subject’s SSL-certificate [203-205]. Moreover, fresh nonce are generated by both parties in both protocols to be included in all exchanged messages to prevent inducing messages from previous sessions, avoid impersonators and to rule out replay attacks.

5.3.1.1 Challenge-Generation Theoretical Measurement

The challenge-generation protocol represents the first phase of the MUAS protocol verification process. It is responsible for forming and encrypting the contents of the QR code image to be displayed on the web-browser as a challenge. In terms of verification, the protocol consists of two roles: the client (browser) and the server (merchant), denoted by $\boldsymbol{B}$ and $\boldsymbol{S}$, respectively. Figure 5.2 depicts the Message Sequence Chart (MSC) [245] of the challenge-generation protocol and shows the exchanged messages among the participants.

Taking the Public Key Infrastructure (PKI) and SSL/TLS handshake protocol into account, both roles have mutually been authenticated via the SSL-certificate exchange. Hence, they shared a secret session key $\boldsymbol{k}(\boldsymbol{B},\boldsymbol{S})$. Moreover, the initial knowledge of each role includes its own public/secret key pair, in addition to the public key of the other party.

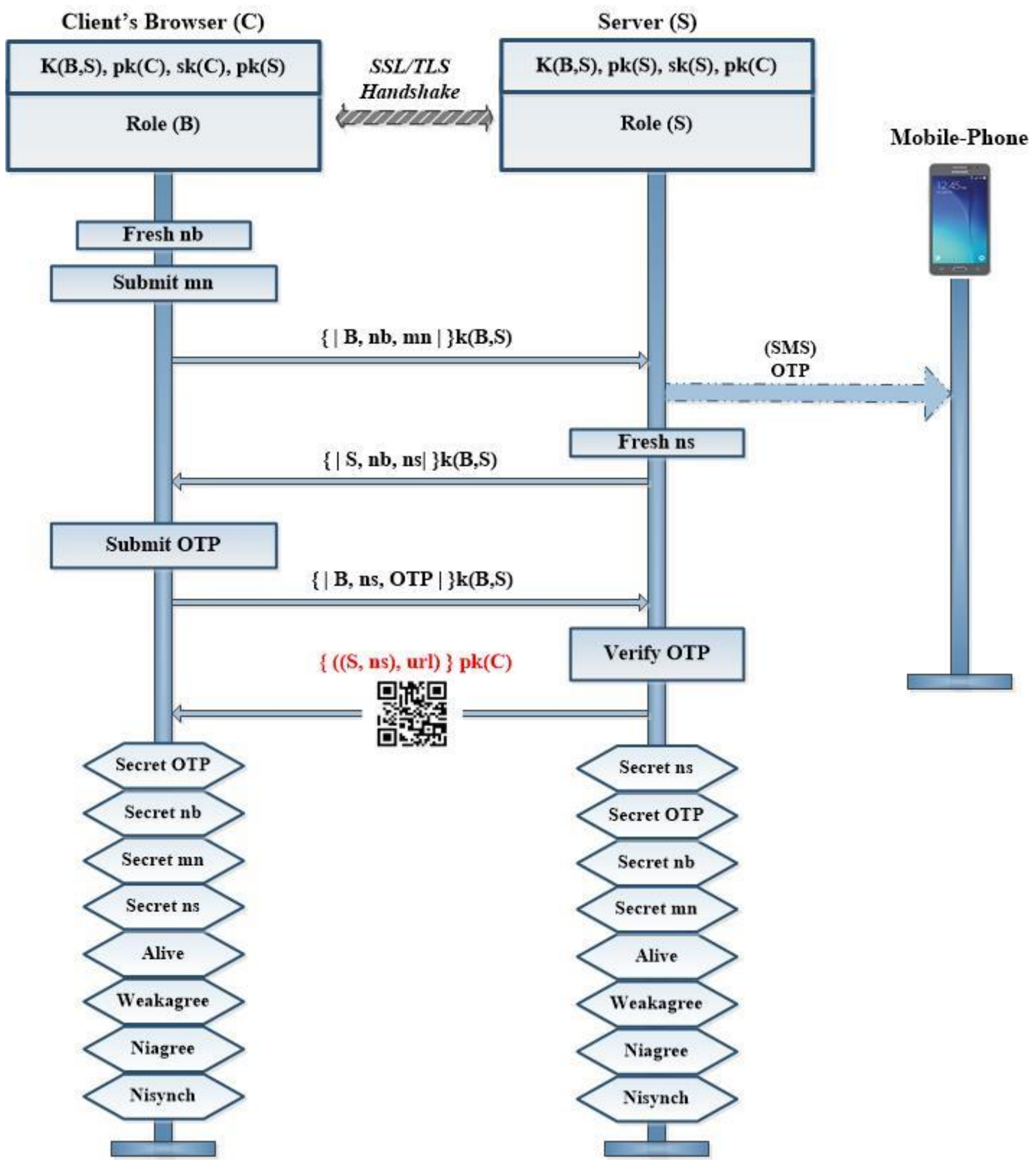


Figure 5.2: Challenge-Generation Protocol

The following steps illustrate and justify the exchanged messages within the challenge-generation protocol, along with their contents and algorithms used.

$$\boldsymbol{C-G\_Message1}: (\,(\boldsymbol{B, nb, mn})_{\boldsymbol{k(B,S)}}) \longrightarrow \boldsymbol{S}$$

Whenever a client ***B*** initiates a connection with a server ***S*** via the browser, browser ***B*** forms the initiating message of the challenge-generation protocol to be sent to the server ***S***, along with the required information.

The message is composed of three elements: an identity ($\boldsymbol{B}$), a fresh browser nonce ($\boldsymbol{nb}$) and the requested mobile number ($\boldsymbol{mn}$), which represents the foundation on which the rest of the protocol is built.

The $\boldsymbol{nb}$ is sent back and forth within the first two messages of the challenge-generation protocol to:

- Prevent Man-in-the-Middle (MITM) attacks via performing an $\boldsymbol{nb}$ handshake, and
- Prevent an adversary from impersonating the server.

$\boldsymbol{C-G\_Message2}$: $\left((\boldsymbol{S}, \boldsymbol{nb}, \boldsymbol{ns})_{\boldsymbol{k(B,S)}} \right) \xrightarrow{\hspace{4em}} \boldsymbol{B}$

Server $\boldsymbol{S}$ decrypts the received $\boldsymbol{C-G\_Message1}$ for recovering and exploiting its contents with regard to generating the complementary $\boldsymbol{C-G\_Message2}$, thus serving the purpose of ensuring server legitimacy via the nonce handshake. Once decrypted, $\boldsymbol{S}$ verifies the user's registration by validating the recovered $\boldsymbol{mn}$ against the *Mobile_Number* field (primary key) within the $Registered\_Clients\,Table$. Upon registration confirmation, $\boldsymbol{S}$ validates the recovered identity $\boldsymbol{B}$ against the SSL-certificate of browser $\boldsymbol{B}$ received during the SSL/TLS handshake protocol. Thus, a new record for client $\boldsymbol{B}$ is created within the $\boldsymbol{Client-Table}$, where all subsequent session variables are saved in this record to be used within the response-generation protocol. Following record $\boldsymbol{B}$ creation, Server $\boldsymbol{S}$ generates a fresh nonce ($\boldsymbol{ns}$) to be sent back and forth within the second and third messages of the protocol, including a server nonce $\boldsymbol{ns}$ within the protocol messages to satisfy the following:

- Prevent MITM attacks via performing an $\boldsymbol{ns}$ handshake, and
- Prevent a client from submitting a message from a previous session, and
- Avoid malicious users impersonating the client.

Next, server $\boldsymbol{S}$ saves the $\boldsymbol{ns}$ in the $Client-Table$ within client $\boldsymbol{B}$ record for later verification-use in step $\boldsymbol{R-G\_Message4}$ within the response-generation protocol.

Eventually, the second message of the protocol is formed by server $\boldsymbol{S}$ and sent to browser $\boldsymbol{B}$. The message is composed of the subject's identity ($\boldsymbol{S}$), fresh server nonce ($\boldsymbol{ns}$) and the recovered browser nonce ($\boldsymbol{nb}$)**,** which is sent back to its originator, representing the crucial step in ensuring the legitimacy of the $\boldsymbol{C-G\_Message2}$ originator.

Upon sending $\boldsymbol{C-G\_Message2}$, $\boldsymbol{S}$ generates an $\boldsymbol{OTP}$ and sends it via SMS to the mobile phone by exploiting the recovered mobile number $\boldsymbol{mn}$ from $\boldsymbol{C-G\_Message1}$. The SMS-OTP will be submitted later as a means to ensure legitimacy of the submitted mobile number $\boldsymbol{mn}$.

$$\boldsymbol{C-G\_Message3}:\ \left(\ (\boldsymbol{B},\boldsymbol{ns},\boldsymbol{OTP})_{\boldsymbol{k(B,S)}}\right) \xrightarrow{\qquad\qquad} \boldsymbol{S}$$

Browser $\boldsymbol{B}$ decrypts the received $\boldsymbol{C-G\_Message2}$ for recovering and verifying the $\boldsymbol{nb}$ handshake, thus ensuring server legitimacy. Once decrypted, $\boldsymbol{B}$ validates the recovered identity $\boldsymbol{S}$ against the SSL-certificate of server $\boldsymbol{S}$ received during the SSL/TLS handshake protocol.

Eventually, the third message of the protocol is formed by browser $\boldsymbol{B}$ and sent to server $\boldsymbol{S}$. The message comprises the subject's identity ($\boldsymbol{B}$), received $\boldsymbol{OTP}$ and the recovered server nonce ($\boldsymbol{ns}$), which is sent back to its originator, representing the crucial step in ensuring the legitimacy of $\boldsymbol{C-G\_Message3}$ originator. Furthermore, the recovered server nonce ($\boldsymbol{ns}$) is included in the third message to ensure the legitimacy of the $\boldsymbol{C-G\_Message3}$ originator, in case the OTP has been eavesdropped.

$$\boldsymbol{C-G\_Message4}:\ \left(\ ((\boldsymbol{S},\boldsymbol{ns}),\boldsymbol{url})_{\boldsymbol{pk(B)}}\right) \xrightarrow{\qquad\qquad} \boldsymbol{B}$$

Server $\boldsymbol{S}$ decrypts the received $\boldsymbol{C-G\_Message3}$ for recovering and exploiting its contents with regard to generating $\boldsymbol{C-G\_Message4}$. Once decrypted, $\boldsymbol{S}$ validates the recovered identity $\boldsymbol{B}$ against the SSL-certificate of client $\boldsymbol{B}$ received during the SSL/TLS handshake protocol.

Following $\boldsymbol{B}$'s identity verification, server $\boldsymbol{S}$ verifies the recovered $\boldsymbol{ns}$ and recovered $\boldsymbol{OTP}$ against its original value generated by the server $\boldsymbol{S}$ in step $\boldsymbol{C-G\_Message2}$. After $\boldsymbol{OTP}$ verification, server $\boldsymbol{S}$ obtains the public key $\boldsymbol{pk(C)}$ of client $\boldsymbol{B}$ via PKI, forms the final message of the challenge-generation protocol and encrypts it with $\boldsymbol{pk(C)}$.

Eventually, the QR code image is generated and sent to browser $\boldsymbol{B}$ to be captured by the mobile phone $\boldsymbol{M}$. The QR code encrypted contents comprises the subject's identity ($\boldsymbol{S}$), its original server nonce ($\boldsymbol{ns}$) and the response end-point ($\boldsymbol{url}$).

5.3.1.2 Response-Generation Theoretical Measurement

The response-generation protocol represents the second phase of the MUAS protocol verification process. It complements the first phase of the MUAS, yet in a different communication channel to finally generate the response via a mobile phone application.

In terms of verification, the response-generation protocol comprises two roles: the server and the mobile phone, denoted by $\boldsymbol{S}$ and $\boldsymbol{M}$, respectively. Figure 5.3 depicts the MSC [245] of the response-generation protocol and shows the exchanged messages among the participants.

Taking the PKI and SSL/TLS handshake protocol into account, server $\boldsymbol{S}$ is authenticated via its SSL-certificate exchange. Hence, sharing a secret session key $\boldsymbol{k}(\boldsymbol{M}, \boldsymbol{S})$ with the mobile phone $\boldsymbol{M}$. Moreover, the initial knowledge of each role includes its own public/private key pair, in addition to the public key of the other party.

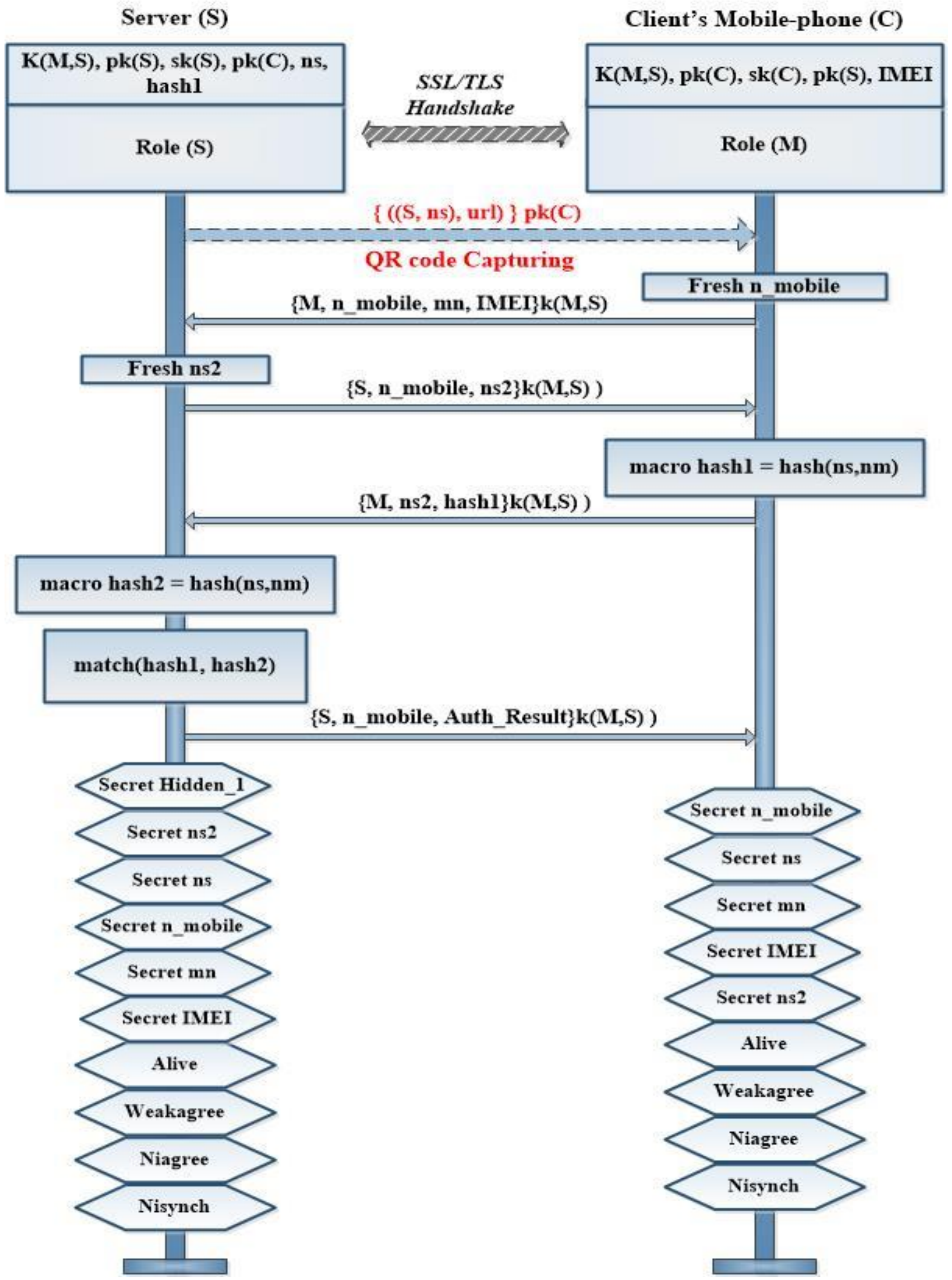


Figure 5.3: Response-Generation Protocol

The following steps illustrate and justify the exchanged messages within the response-generation protocol, along with their contents and algorithms used.

$$\boldsymbol{R-G\_Message1}: (\,(\boldsymbol{M},\ \boldsymbol{n\_mobile},\ \boldsymbol{mn},\ \boldsymbol{IMEI})_{\boldsymbol{k(M,S)}}) \xrightarrow{\qquad\qquad} \boldsymbol{S}$$

Following capture of the QR code, decryption and contents recovering, mobile phone $\boldsymbol{M}$ initiates a connection with server $\boldsymbol{S}$ via the recovered response end-point $\boldsymbol{url}$. Mobile phone $\boldsymbol{M}$ forms the initiating message of the response-generation protocol to be sent to the server $\boldsymbol{S}$ implicitly.

The message comprises four elements: an identity ($\boldsymbol{M}$), a fresh mobile nonce ($\boldsymbol{n\_mobile}$), an extracted mobile number ($\boldsymbol{mn}$), and the extracted IMEI ($\boldsymbol{IMEI}$). The $\boldsymbol{n\_mobile}$ is sent back and forth within the first two messages of the response-generation protocol to:

- Prevent replay attacks via performing an $\boldsymbol{n\_mobile}$ handshake, and
- Avoid malicious users impersonating the server.

$$\boldsymbol{R-G\_Message2}: ((\boldsymbol{S},\ \boldsymbol{n\_mobile}, \boldsymbol{ns2})_{\boldsymbol{k(M,S)}}) \longrightarrow \boldsymbol{M}$$

Server $\boldsymbol{S}$ decrypts the received $\boldsymbol{R-G\_Message1}$ for recovering and exploiting its contents with regard to generating the complementary $\boldsymbol{R-G\_Message2}$, serving the purpose of ensuring server legitimacy. Once decrypted, $\boldsymbol{S}$ verifies the user's registration by validating the recovered $\boldsymbol{mn}$ against the *Mobile_Number* field (primary key) within the $Registered\_Clients\ Table$. Upon registration confirmation, the recovered $\boldsymbol{IMEI}$ is verified against the registered International Mobile Equipment Identity number $\boldsymbol{Reg\_IMEI}$ field within the same registered user in the $Registered\_Clients\ Table$.

Following the $\boldsymbol{mn}$ and $\boldsymbol{IMEI}$ verification, server $\boldsymbol{S}$ generates a fresh nonce ($\boldsymbol{ns2}$) to be sent back and forth within the second and third messages of the protocol, including a server nonce $\boldsymbol{ns2}$ within the protocol messages to satisfy the following:

- Prevent an adversary from impersonating the client,
- Prevent a client from submitting a message from a previous session, and
- Prevent replay attacks via performing an $\boldsymbol{ns2}$ handshake.

Eventually, the second message of the protocol is formed by server $\boldsymbol{S}$ and sent to the mobile phone $\boldsymbol{M}$. The message comprises the subject's identity ($\boldsymbol{S}$), fresh server nonce ($\boldsymbol{ns2}$) and the recovered mobile nonce ($\boldsymbol{n\_mobile}$), which is sent back to its originator, representing the crucial step in ensuring the legitimacy of $\boldsymbol{R-G\_Message2}$ originator.

$\longrightarrow$

$\boldsymbol{R - G\_Message3}$: ($(\boldsymbol{M}, \boldsymbol{ns2}, \boldsymbol{hash1})_{\boldsymbol{k(M,S)}}$) $\boldsymbol{S}$

Mobile phone $\boldsymbol{M}$ decrypts the received $\boldsymbol{R - G\_Message2}$ for recovering and exploiting its contents with regard to generating the complementary $\boldsymbol{R - G\_Message3}$, serving the purpose of ensuring client legitimacy. Once decrypted, $\boldsymbol{M}$ validates the recovered identity $\boldsymbol{S}$ against the SSL-certificate of server $\boldsymbol{S}$ received during the SSL/TLS handshake protocol and against the QR code-recovered server identity ($\boldsymbol{S}$).

Once the recovered mobile nonce ($\boldsymbol{n\_mobile}$) has been verified, mobile phone $\boldsymbol{M}$ appends the $\boldsymbol{n\_mobile}$ with the recovered $\boldsymbol{ns}$ received upon capturing and decoding the QR code contents. Next, a one-way hash is applied on the appended result, thus generating the response:

$$macro\ hash1 = hash(ns, n\_mobile).$$

Eventually, the third message of the protocol is formed by mobile phone $\boldsymbol{M}$ and sent to server $\boldsymbol{S}$. The message comprises the subject's identity ($\boldsymbol{M}$), hash result ($\boldsymbol{hash1}$) and the recovered server nonce $\boldsymbol{n}$ ($\boldsymbol{s2}$), which is sent back to its originator, representing the crucial step in ensuring the legitimacy of the $\boldsymbol{R - G\_Message3}$ originator.

$\boldsymbol{R - G\_Message4}$: ($(\boldsymbol{S}, \ \boldsymbol{n\_mobile}, \boldsymbol{Auth\_Result})_{\boldsymbol{k(M,S)}}$) $\longrightarrow$ $\boldsymbol{M}$

Server $\boldsymbol{S}$ decrypts the received $\boldsymbol{R - G\_Message3}$ for recovering and exploiting its contents with regard to validating the received response. Once decrypted, $\boldsymbol{S}$ validates the recovered identity $\boldsymbol{M}$ against the user identity field (primary key) within the $Client - Table$.

Once the recovered server nonce $\boldsymbol{ns2}$ has been verified, the server will $\boldsymbol{'LOOKUP'}$ the record $\boldsymbol{M}$ in the $Client - Table$ to find the server nonce $\boldsymbol{ns}$ field and append it with the recovered $\boldsymbol{n\_mobile}$ from the $\boldsymbol{R - G\_Message1}$. Next, a one-way hash ($\boldsymbol{hash2}$) is applied to the appended result to finally be compared against the recovered $\boldsymbol{hash1}$ from $\boldsymbol{R - G\_Message3}$:

$$macro\ hash2 = hash(ns, n\_mobile);$$

Finally, the fourth message is formed by server $\boldsymbol{S}$ and sent to mobile phone $\boldsymbol{M}$. The message comprises the subject's identity ($\boldsymbol{S}$), authentication result ($\boldsymbol{Auth\_Result}$) and the recovered mobile nonce ($\boldsymbol{n\_mobile}$) that is utilised to prevent replay attack and to ensure the legitimacy of the message originator.

5.3.2Implementation Performance Measurement

When implementing the challenge- and response-generation protocols, it was taken into consideration that security attributes are attached to the data required by the authentication system to be exchanged via the established SSL/TLS communication channel. This affected the computational and communication costs in terms of the number of exchanged messages between participants and the size of each message in bytes.

5.3.2.1 Computational Cost Measurement

The constraint of developing a fast and secure authentication system has been considered while designing the protocol; hence, the protocol comprises security-related functions along with simple and light-weight functions.

With regard to satisfying the security properties, random number generators were executed twice in both protocols by both participants to represent nonce for securing the protocol against impersonators and malicious users, as well as to prevent replay attacks. It was also executed within the challenge-generation protocol for generating an OTP. Moreover, light-weight hash functions were executed once in each protocol, where the first hash is used as a reference for verifying the received response, while the second hash represents the generated response. In terms of execution speed, all exchanged messages were symmetrically encrypted, as expensive public key encryption was used only once in the challenge-generation protocol to be decrypted in the response-generation protocol. Table 5.1 shows a comparison between our protocol and the most related protocols in terms of the number of performed operations in each protocol.

Table 5.1: Performance comparison between MUAS protocol and the most related work

	Phase	**Hash**	**HMAC**	**Rand No. Gen.**	**Shared Key Gen.**	**P-key Encrypt**	**S-key Encrypt**	**QR code Gen.**
	Challenge Gen.	-	-	2 (nonce) 1 (OTP)	-	1	Comm. encrypted	1

MUAS Protocol	Response Gen.	1	-	2 (nonce)	-	-	Comm. encrypted	-
	Response Verify	1	-	-	-	-	Comm. encrypted	-
OTP calculated from QR code [200]	Challenge Gen.	-	-	2 (random value, random length)	-	1	Comm. encrypted	1
	Response Gen.	1	-	-	-	-	Comm. encrypted	-
	Response Verify	1	-	-	-	-	Comm. encrypted	-
Snap2Pass [201]	Account Creation (Challenge Gen.)	-	-	-	1	-	Comm. encrypted	1
	Challenge Gen.	-	-	1	-	-	Comm. encrypted	1
	Response Gen.	-	1	-	-	-	Comm. encrypted	-
	Response Verify	-	1	-	-	-	Comm. encrypted	-
2-clickAuth [130]	Account Creation (Challenge Gen.)	-	-	-	1	1	-	1
	Challenge Gen.	-	-	1	-	-	-	1
	Response Gen. (QR code)	-	1	-	-	-	-	1
	Response Verify	-	1	-	-	-	-	-

From Table 5.1, our protocol generates the most random numbers, representing the fresh nonce that is included within exchanged messages as a means to satisfy security properties, which is required for securing protocols [41]. Although the computational cost of our protocol is higher in terms of random number generation than the work in [200], Snap2Pass [201] and 2-clickAuth [130]; however, a trade-off was made with respect to protocol security.

In terms of energy consumption, a comprehensive analysis of the energy requirements of a wide range of cryptographic algorithms has been presented in [246]. The study showed that energy consumption of the hash and MAC algorithms consumes the least energy of all cryptographic algorithms. Therefore, our protocol employed the simple and light-weight hash algorithm in the response generation and verification, which is similar to the Snap2Pass [201] and 2-clickAuth [130] that employed the computation-intensive hash-based MAC (HMAC). The only difference is that the HMAC requires an established secret key between participants, which is accomplished within the registration phase in theSnap2Pass and 2-clickAuth. Whereas the MUAS participants do not share any secret keys, thus a hash algorithm is used.

On the other hand, our protocol, [200], and the 2-clickAuth [130] employed the expensive public key encryption once to secure the QR code contents; whereas the QR code contents of Snap2Pass [201] are not encrypted, which is considered a weak point in the account creation phase due to the risk of peepers. Furthermore, exchanged messages within all proposed protocols, except for 2-clickAuth [130], are symmetrically encrypted due to executing over the SSL/TLS handshake protocol.

5.3.2.2 Communication Cost Measurement

Communication costs refer to the number of exchanged messages between the client and the server in both protocols, along with the size of each message in bytes. The size of each message has been calculated according to the size of the parameters and the cryptographic algorithms' outputs, which are as follows:

- **User ID**: represented by the 64 bytes DN, which is used to identify the owner of the certificate. The DN is made up of: common name, organisational unit, organisation, locality, state or province, and country name [205].
- **Nonce**: cryptographic nonce is of size 32 bytes (256-bits). It is generated using Fortuna PRNG.
- **Hash-256**: the function's maximum input size is $2^{64} - 1$, and it generates a hash of fixed length that is 32 bytes (256-bits).
- **Symmetric key**: the established shared secret key used for symmetrical encryption of size 128-bits.
- **RSA key**: used for QR code encryption of size 245 bytes (1960-bits).

Table 5.2 compares our protocol and the most related protocols in terms of the number and size of exchanged messages in each protocol.

Table 5.2: Comparison of communication costs between MUAS protocol and other relevant protocols

	Phase	No. of Messages	Size of Messages
MUAS Protocol	Challenge Gen.	4	[Msg1- Msg4: 172/ 216/ 172/ bytes] (SSL) [RSA encrypted QR code: 344 bytes]
	Response Gen.	4	[Msg1- Msg4: 216/ 216/ 216/ 172/ bytes]
	Challenge Gen.	2	[PKI_UserID: 108 bytes] (SSL)

OTP calculated from QR code [200]			[RSA encrypted QR code: 172 bytes]
	Response Gen.	1	[OTP: 24 bytes] (SSL)
Snap2Pass [201]	Account Creation	2	[Username: 24 bytes] (SSL) [QR code encoded challenge: 71 bytes]
	Challenge Gen.	2	[Session ID, Nonce: 44 bytes] (SSL) [QR code encoded challenge: 51 bytes]
	Response Gen.	2	[QR code encoded challenge: 51 bytes] [Authentication result: 5 bytes]
2-clickAuth [130]	Account Creation	1	[RSA encrypted key barcode: 172 bytes]
	Challenge Gen.	1	[QR code encoded challenge: 128-bit]
	Response Gen.	1	[QR code encoded response: 128-bit]

From Table 5.2, our protocol exceeds other protocols in terms of the number of exchanged messages and their sizes. This can be justified by the objective of establishing a secure authentication system against adversaries and replay attacks. Cryptographic security attributes are included within all messages, along with the identities of the sender as a means for identity verification.

Another reason for having more messages than Snap2Pass [201], 2-clickAuth [130] and [200] lies within discarding the first challenge-generation protocol to initiate a new and detached protocol for response generation. The second protocol follows the same security measures for developing a secure protocol that satisfies the required security properties for adversary prevention. Therefore, more messages are involved within the protocol that includes cryptographic nonce and identities along with the generated response.

5.3.2.3 Execution Time Measurement

Measuring the execution times for both phases of the MUAS is based on the formation and cryptography of each message within the protocol. Random runs have been executed multiple times to calculate the average run time for each protocol. Tables 5.3 and 5.4 show the total execution times for each run and the average run time for the entire protocol. For the participants, it also shows the average time to form and encrypt each message being sent within the protocol as well as the average run time to decrypt the received message.

Table 5.3: Execution Time of Challenge Generation Protocol

Challenge-Generation		Form-Encrypt Message1	Decrypt Message1 & Form-Encrypt Message2	Decrypt Message2 & Form-Encrypt Message3	Decrypt Message3 & Form-Encrypt Message4	Total Run Time (sec)
Run1	Client	0.016	-	0.016	-	13.64
	Server	-	0.015	-	0.047	
Run2	Client	0.016	-	0.015		14.14
	Server	-	0		0.063	
Run3	Client	0.031		0.016		15.18
	Server		0.016		0.078	
Run4	Client	0.016		0.015		14.07
	Server		0.016		0.063	
Run5	Client	0.015		0.015		12.15
	Server		0.016		0.062	
Run6	Client	0.016		0		11.75
	Server		0		0.062	
Run7	Client	0.016		0.016		12.32
	Server		0.015		0.063	
Run8	Client	0.016		0		13.98
	Server		0.016		0.047	
Run9	Client	0.016		0.016		16.54
	Server		0.015		0.062	
Run10	Client	0.016		0.016		14.28
	Server		0.015		0.063	
Average run time for forming messages (sec)		0.02	0.01	0.01	0.06	13.81

Table 5.3 contains four columns. The first column, $\boldsymbol{Form - Encrypt\ Message1}$, represents the process of forming the first message within the challenge generation protocol. This includes receiving user-input, extracting the sender's identity, generating the cryptographic nonce, and encrypting the message. The rest of the columns include the processes of decrypting the received messages to form the subsequent messages. The average execution times for $\boldsymbol{Message1}$, $\boldsymbol{Message2}$, $\boldsymbol{Message3}$, and $\boldsymbol{Message4}$ are ***0.02***, ***0.01***, ***0.01***, and ***0.06*** seconds, respectively. The average run time for the entire protocol is ***13.81*** seconds.

Table 5.4: Execution Time of Response Generation Protocol

Response-Generation		Form-Encrypt Message1	Decrypt Message1 & Form-Encrypt Message2	Decrypt Message2 & Form-Encrypt Message3	Decrypt Message3 & Form-Encrypt Message4	Total Run Time (sec)
Run1	Client	0.0023		0.0040		0.1525
	Server		0.06		0.03	
Run2	Client	0.0045		0.0024		0.1985
	Server		0.06		0.03	
Run3	Client	0.0045		0.0023		0.1331
	Server		0.05		0.03	
Run4	Client	0.0023		0.0023		0.0862
	Server		0.03		0.02	
Run5	Client	0.0045		0.0022		0.1502
	Server		0.06		0.03	
Run6	Client	0.0023		0.0024		0.1318
	Server		0.05		0.03	
Run7	Client	0.0024		0.0024		0.1498
	Server		0.06		0.03	
Run8	Client	0.0026		0.0025		0.1042
	Server		0.05		0.02	
Run9	Client	0.0049		0.0034		0.1266
	Server		0.05		0.03	
Run10	Client	0.0023		0.0025		0.1636
	Server		0.06		0.05	
Average run time for forming messages (sec)		0.0033	0.0529	0.0026	0.0298	0.1397

Table 5.4 also contains four columns. The first column, $\boldsymbol{Form - Encrypt\ Message1}$, represents the process of forming the first message in the response generation protocol. Unlike the challenge generation protocol that requires the user to submit their mobile number and one-time-password, this protocol does not require any user-input. Thus, the challenge generation phase requires more time, as it is an input-based protocol that requires the client to submit information.

5.4 MUAS Adversary Model

Based on the described MUAS model in Chapter 1, this section presents the adversary model of our thesis. It is assumed that trusted agents represent both the authentication server and the mobile device of an honest end-user. Thus, an attacker cannot access any secret information within both parties. However, it is assumed that the adversary has complete control over the user's browser, operating on the internet browser application-level, thus having the potential to learn and manipulate any data entered, received or sent through the compromised browser. In addition, the adversary can tamper with data of an outgoing-form that is submitted to the server. With the adversary residing in the browser, any user input and security attributes generated by the browser or received from the server with respect to the protocol security are not secure.

Although SSL/TLS symmetrically encrypts exchanged messages, it has no effect on securing the data from the adversary. The data is encrypted and decrypted in the transport layer, which is below the windows socket application programming interface (API). Thus, the adversary modifies the incoming data after decryption and the outgoing data before encryption [1]. The adversary can simply circumvent the encryption and decryption within the SSL/TLS protocol. Furthermore, the security measures of the public key infrastructures can be circumvented as well [14]. In a nutshell, it is assumed the adversary has compromised the client PC (browser), thus able to manipulate all incoming and outgoing messages.

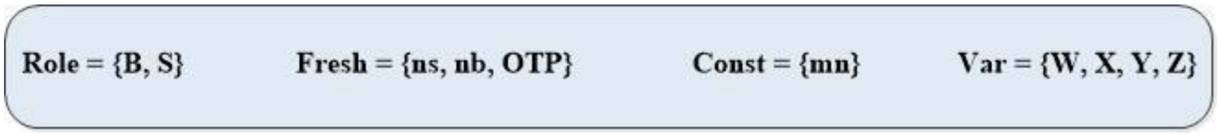
5.5 MUAS Protocol Specification

Considering the required security properties of a protocol, the behaviour of each role within both the challenge- and response-generation protocols is mandatory. Therefore, specification of both protocols are provided in this section, which delivers a description of each protocol in an abstract manner. The specification includes the initial knowledge of the protocol that is required for role execution along with function declarations, fresh values generation, and global constants and variables definitions [41].

The following subsections describe both protocols in an expressive specification language, paving the way for justifying and analysing the security properties.

5.5.1Challenge-Generation Protocol Specification

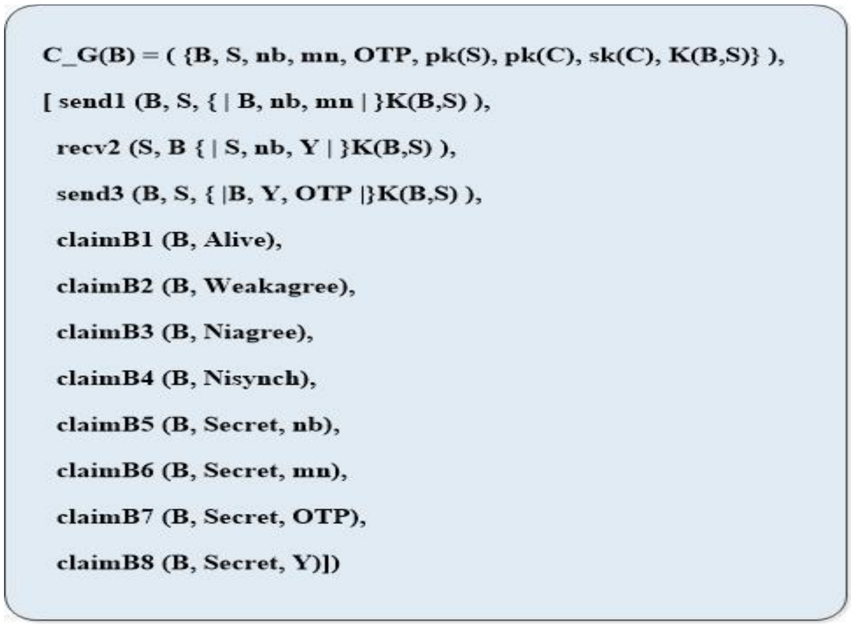
The initial challenge-generation protocol comprises two roles: the browser ($\boldsymbol{B}$) and the server ($\boldsymbol{S}$), where each role is specified as a sequence of events by using static descriptions, such as send and receive messages. The general initial knowledge required for the challenge-generation ($\boldsymbol{C\_G}$) protocol execution is presented in List 5.1.



Role = {B, S} **Fresh = {ns, nb, OTP}** **Const = {mn}** **Var = {W, X, Y, Z}**

List 5.1: General Initial Knowledge of Challenge-Generation.

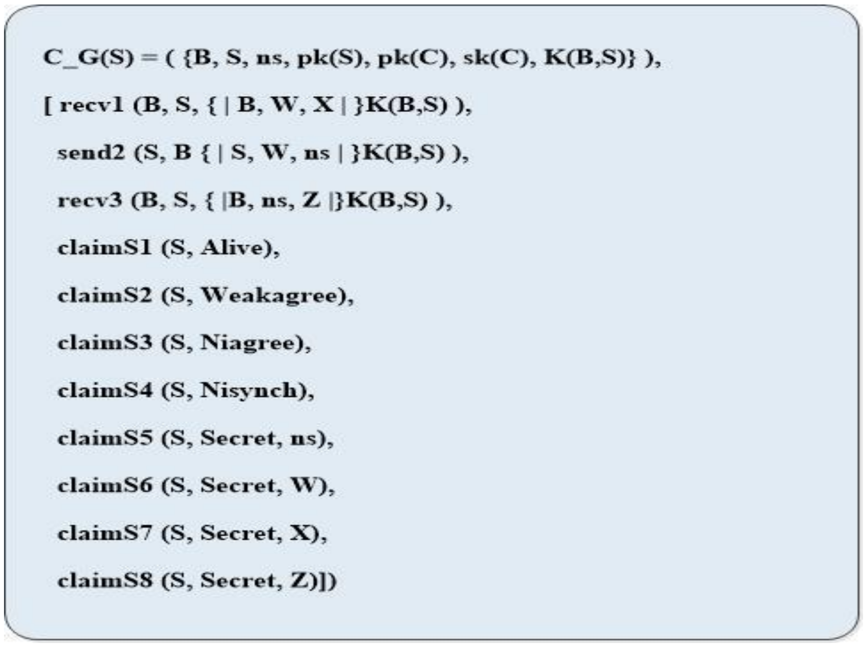
The browser role specification is presented in List 5.2, which models the initiating browser-side $\boldsymbol{B}$ role (client-side) of the challenge-generation ($\boldsymbol{C\_G}$) protocol. Alongside the specification, initial knowledge of the browser role is included, specifying its freshly generated nonce, along with its public/secret key pair, public key of the corresponding party and their shared secret key.



```
C_G(B) = ( {B, S, nb, mn, OTP, pk(S), pk(C), sk(C), K(B,S)} ),
[ send1 (B, S, { | B, nb, mn | }K(B,S) ),
  recv2 (S, B { | S, nb, Y | }K(B,S) ),
  send3 (B, S, { |B, Y, OTP |}K(B,S) ),
  claimB1 (B, Alive),
  claimB2 (B, Weakagree),
  claimB3 (B, Niagree),
  claimB4 (B, Nisynch),
  claimB5 (B, Secret, nb),
  claimB6 (B, Secret, mn),
  claimB7 (B, Secret, OTP),
  claimB8 (B, Secret, Y)])
```

List 5.2: Specification of Server Role (B) in the Challenge-Generation Protocol.

The corresponding server role specification is presented in List 5.3, which models server $\boldsymbol{S}$ role (merchant-side) of the challenge-generation ($\boldsymbol{C\_G}$) protocol. Alongside the specification, initial knowledge of the server role is included, specifying its freshly generated nonce, its input, along with its public/secret key pair, public key of the corresponding party and their shared secret key.

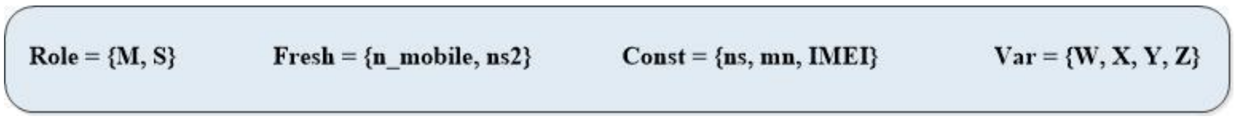


```
C_G(S) = ( {B, S, ns, pk(S), pk(C), sk(C), K(B,S)} ),
[ recv1 (B, S, { | B, W, X | }K(B,S) ),
  send2 (S, B { | S, W, ns | }K(B,S) ),
  recv3 (B, S, { |B, ns, Z |}K(B,S) ),
  claimS1 (S, Alive),
  claimS2 (S, Weakagree),
  claimS3 (S, Niagree),
  claimS4 (S, Nisynch),
  claimS5 (S, Secret, ns),
  claimS6 (S, Secret, W),
  claimS7 (S, Secret, X),
  claimS8 (S, Secret, Z)])
```

List 5.3: Specification of Browser Role (S) in the Challenge-Generation Protocol.

5.5.2Response-Generation Protocol Specification

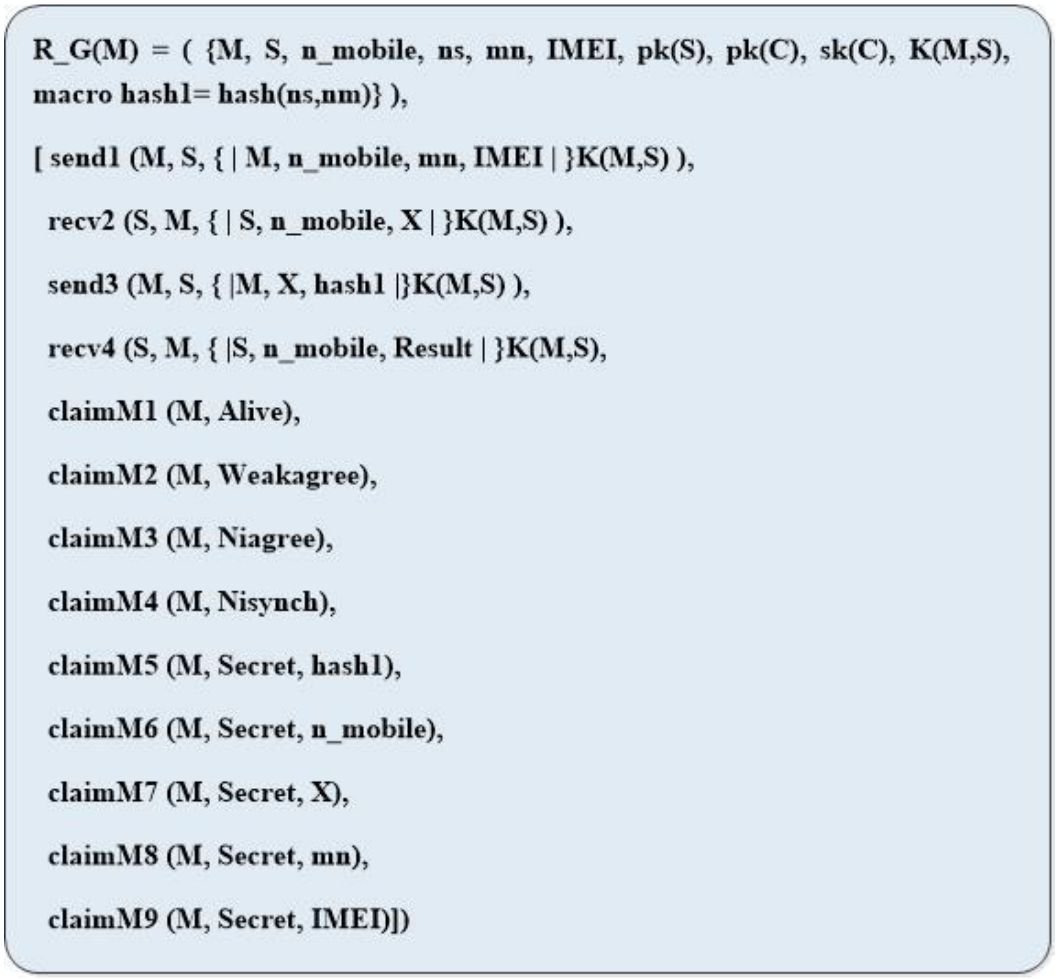
The response-generation protocol comprises two roles: the server ($\boldsymbol{S}$) and the mobile-phone ($\boldsymbol{M}$). Along with the send and receive sequence events, the specification includes internal computations that are represented by pattern match events. The general initial knowledge required for response-generation ($\boldsymbol{R\_G}$) protocol execution is presented in List 5.4.



```
Role = {M, S}        Fresh = {n_mobile, ns2}        Const = {ns, mn, IMEI}        Var = {W, X, Y, Z}
```

List 5.4 Response-Generation Initial Knowledge

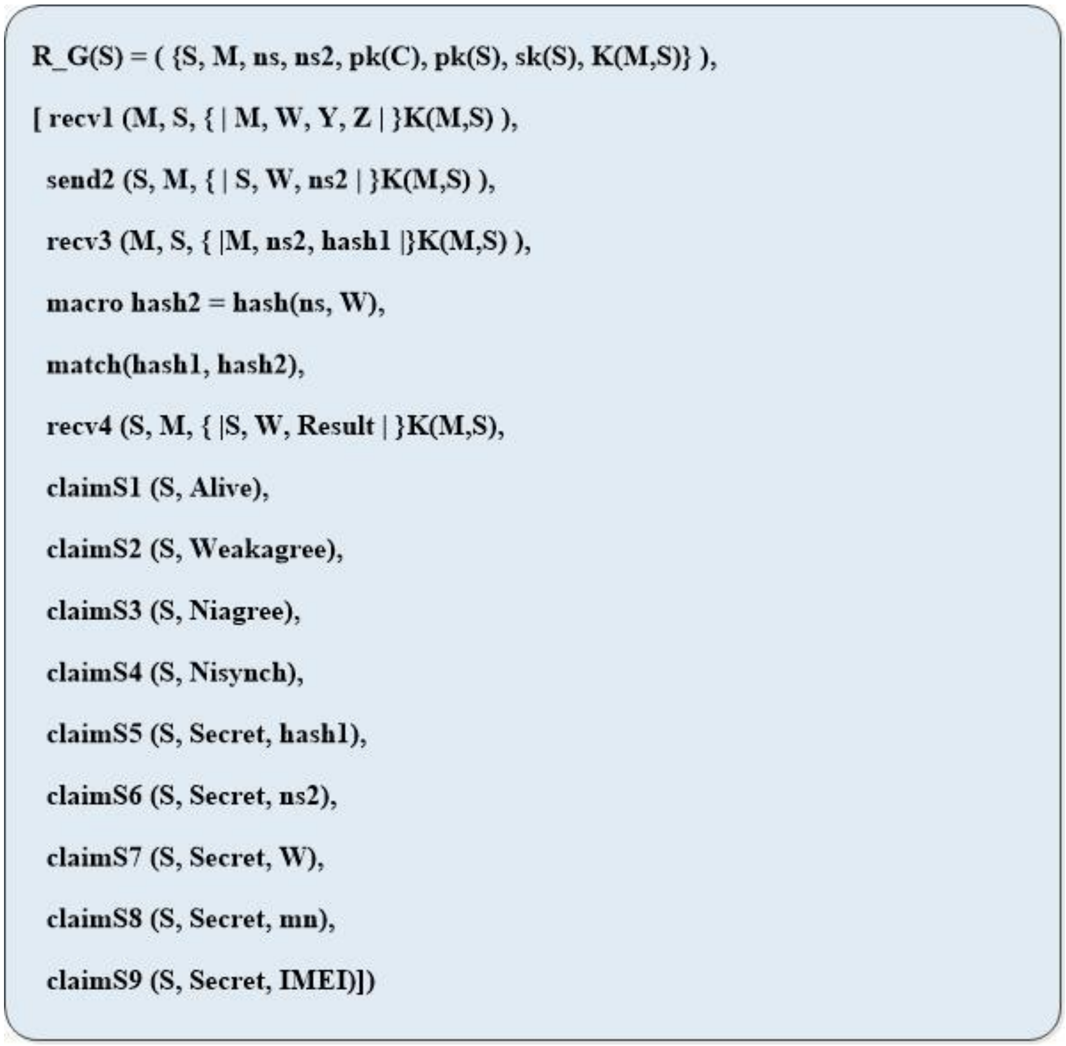
The mobile phone specification is presented in List 5.5, which models the mobile phone ($\boldsymbol{M}$) (client-side) role of the response-generation ($\boldsymbol{R\_G}$) protocol. Alongside the specification, initial knowledge of the mobile phone role is included, specifying its freshly generated nonce, its global constant, along with its public/secret key pair, public key of the corresponding party and their shared secret key. In addition, a declared hash function that represents the basis for the response-generation protocol is included.



```
R_G(M) = ( {M, S, n_mobile, ns, mn, IMEI, pk(S), pk(C), sk(C), K(M,S),
macro hash1= hash(ns,nm)} ),
[ send1 (M, S, { | M, n_mobile, mn, IMEI | }K(M,S) ),
  recv2 (S, M, { | S, n_mobile, X | }K(M,S) ),
  send3 (M, S, { |M, X, hash1 |}K(M,S) ),
  recv4 (S, M, { |S, n_mobile, Result | }K(M,S),
  claimM1 (M, Alive),
  claimM2 (M, Weakagree),
  claimM3 (M, Niagree),
  claimM4 (M, Nisynch),
  claimM5 (M, Secret, hash1),
  claimM6 (M, Secret, n_mobile),
  claimM7 (M, Secret, X),
  claimM8 (M, Secret, mn),
  claimM9 (M, Secret, IMEI)])
```

List 5.5: Specification of Mobile Phone Role (*M*) in the Response-Generation Protocol

The corresponding server role specification is presented in List 5.6, which models the server ($\boldsymbol{S}$) role of the response-generation ($\boldsymbol{R\_G}$) protocol. Alongside the specification, initial knowledge of the server role is included, specifying its freshly generated nonce, its global constant, along with its public/secret key pair, public key of the corresponding party and their shared secret key.



```
R_G(S) = ( {S, M, ns, ns2, pk(C), pk(S), sk(S), K(M,S)} ),
[ recv1 (M, S, { | M, W, Y, Z | }K(M,S) ),
  send2 (S, M, { | S, W, ns2 | }K(M,S) ),
  recv3 (M, S, { |M, ns2, hash1 |}K(M,S) ),
  macro hash2 = hash(ns, W),
  match(hash1, hash2),
  recv4 (S, M, { |S, W, Result | }K(M,S),
  claimS1 (S, Alive),
  claimS2 (S, Weakagree),
  claimS3 (S, Niagree),
  claimS4 (S, Nisynch),
  claimS5 (S, Secret, hash1),
  claimS6 (S, Secret, ns2),
  claimS7 (S, Secret, W),
  claimS8 (S, Secret, mn),
  claimS9 (S, Secret, IMEI)])
```

List 5.6: Specification of Server Role (*S*) in the Response-Generation Protocol

5.6 MUAS Protocol Execution and Verification Results

Upon specifying all the details of both protocols, which include justifying the corresponding messages along with their contents, introducing the possible adversary model and presenting the formal protocol description, this section maps all the previously discussed details into the Scyther verification tool for execution and generation of the verification results for both protocols.

In the previous section, static description of how both protocols should behave were formalised. Upon protocol execution, dynamic aspects are generated. One of which is the agent model that is responsible for executing roles of the protocol. An agent model is based upon closed world assumption, where an honest agent executes the protocol based on the static protocol description. In addition, an agent may execute any number of roles in parallel.

Executing a role turns a role specification into a run. This process is called instantiation. Whenever a role is instantiated, a unique run identifier is assigned to each run. Furthermore, role names are bind to the names of actual agents executing the roles, and fresh values generated by different runs are appended to the identifiers of their runs. Therefore, the set of run terms are different than the set of terms in role specification, where fresh values, roles, and variables that are local to a run are unique via appending them with its run identifier.

The verification mechanism generates the results through translating the protocol description into a Security Protocol Description Language (SPDL) as a means for defining the tool's input. An automated verification and a representative analysis of the security claim events are produced as its output, which will be discussed and analysed with respect to the required security properties.

5.6.1Security Properties

Security properties are an indispensable part of all secure protocols. Knowing the properties to be satisfied is fundamental for a protocol to be considered [41, 247]. According to Roscoe [248], security properties are categorised into intentional and extensional properties. Intentional security properties are motivated by the protocol structure, whereas extensional security properties concern the accomplishment of the protocol.

This section discusses and analyses the essential properties that are verified by a formal verification tool. Secrecy and several forms of authentication are the basic verified properties, formalised through integration into the protocol specification via claim events.

5.6.1.1 Secrecy/Confidentiality

The secrecy security property ensures the confidentiality of any specified role term from being exposed to an adversary, and that they are available to authorised parties only. This property is satisfied if, for each assigned run, where its role is mapped to an honest agent communicates with other honest agents [41].

5.6.1.2 Authentication

Based on the employed Scyther verification tool, three forms of authentication are discussed and analysed in this section. The authentication forms are: aliveness, synchronisation and message agreement.

5.6.1.2.1 Aliveness

The aliveness security property is an intentional property that assures an honest intended communicating partner has executed some events. This property is satisfied whenever an agent executes a role specification up to the claim event; hence, its intended communicating partner has been running the protocol and executed some events, and therefore it is considered *Alive* [41]. There are four forms of aliveness: *weak alive*, *weak alive in the correct role*, *recent alive* and *recent alive in the correct role* [41].

If agents execute their roles up to their claim events, and their intended communicating partners are considered *honest*, then the intended communicating partner has executed an event; thus, the w*eak aliveness* property is satisfied [41, 247]. However, satisfying the *weak aliveness* property is not enough, as it only verifies that the intended communicating partner is alive [41, 247], while he/she may not be running within the same protocol [249]. Plus, the intended communicating partner might not have been running the protocol recently [249]. Therefore, it is preferable to ensure the intended communicating partner is running within the same protocol, which satisfies *weak aliveness in the correct role* [41].

However, *weak aliveness in the correct role* does not say whether an event has been executed before, after or during the run that makes the claim [41, 247]. Thus, an interpretation of recentness can be given through the *recent aliveness* property [247]. This property is satisfied within a protocol if the intended communicating partner has recently sent messages before the claim events, and after other events within the same run in which the claim event occurs [41]. Similarly to *weak aliveness in the correct role*, it is preferable to ensure the intended communicating partner is running within the same protocol, which satisfies *recent aliveness in the correct role [41]*. Furthermore, the *recent aliveness* property can be extended via the *weak agreement* property, which assures the intended communicating partner has agreed to running the protocol with the initiator.

Figure 5.4 shows the Aliveness hierarchy, where any protocol satisfying *recent aliveness* also satisfies *weak aliveness* [41], and any protocol satisfying *recent aliveness in the correct role* also satisfies *weak aliveness in the correct role* [41].

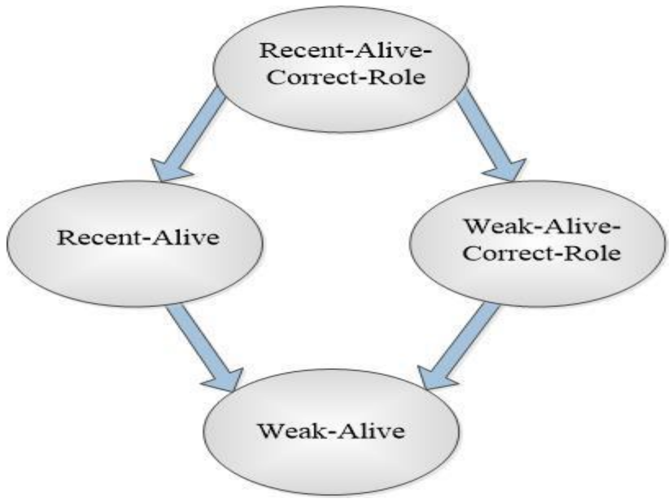


Figure 5.4: Aliveness hierarchy

5.6.1.2.2 Synchronisation

The synchronisation security property is a strong intentional property, requiring all communication messages along with their contents to occur exactly as specified by the protocol description, with respect to the order of communicating events [41, 250, 251]. As agents are capable of executing multiple roles, and a different agent can execute the same role, it is mandatory to identify which run performs which role. Thus, synchronisation heavily depends upon a cast function that maps each claim and each role into a run identifier [41, 247, 252]. There are two forms of synchronisation: non-injective synchronisation and injective synchronisation [41, 247, 252]. Figure 5.5 shows the synchronisation hierarchy, where any protocol satisfying injective synchronisation also satisfies non-injective synchronisation [41].

According to the non-injective synchronisation definition in [41, 247, 252], this property is satisfied when correspondence between the structures in the trace-level and the protocol descriptions are met.

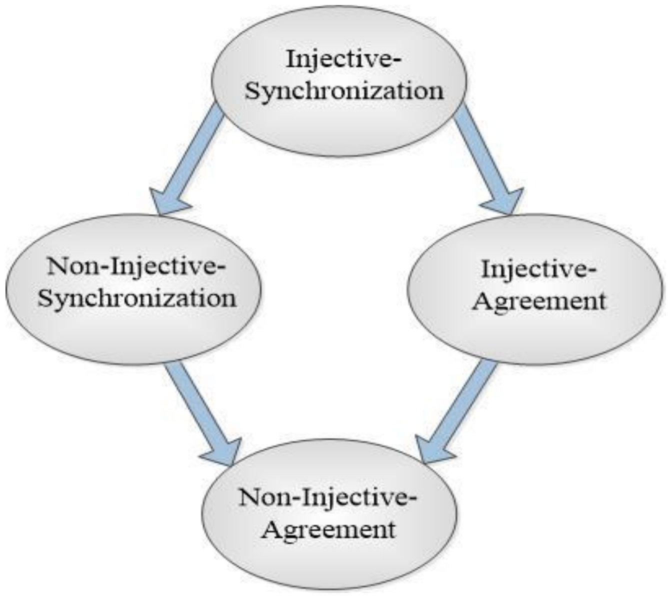


Figure 5.5: synchronisation hierarchy

5.6.1.2.3 Message Agreement/Integrity

The agreement security property is an extensional property that relates to the effect of the protocol after execution [41, 247]. It requires that all contents of the corresponding messages have the exact contents as specified by the protocol description. Upon successful protocol execution, the participants agree upon the contents of the variables, as specified by the protocol description [41, 241, 247, 250]. The agreement property is very similar to synchronisation, except for placing less restrictions on event ordering. Thus, there are two forms of agreement: non-injective agreement and injective agreement [41]. According to figure 5.5, any protocol satisfying injective synchronisation also satisfies injective and non-injective agreement [41].

5.6.2 Scyther Verification Tool

There are several security protocol verification tools based on the formal method approach. A formal method approach assumes perfect cryptography [253, 254], verifies a secure protocol within the existence of an adversary and provides representative analysis to help understand how the attack violates the proposed protocol to be rectified [254]. Based on the comparative analysis of the formal method model checking tool, it was found that Scyther is considered one of the most efficient tools among many formal model checking tools, in terms of public availability, flawed protocol falsification, bounded and unbounded verification, and assured termination [254]. This is in addition to its ability of providing an automated claim verification and generating a representative finite set of traces, which includes the anticipated correct behaviour and a set of all possible attack violations [253].

In terms of verification speed, Scyther can run the protocol for an unbounded number of sessions in less than one second. It also offers the possibility of bounded model checking, where inability to reach an unbounded correctness exists, thus guaranteeing protocol termination [241, 255, 256]. Scyther allows protocol designers to spot protocol attack violations through complete characterisation of each role, thus generating representative and self-explanatory attack graphs for each claim [253-255]. Therefore, the Scyther verification tool was found to be the most appropriate for verifying the required security properties; thus, it was considered in this chapter for verifying the proposed protocols.

5.6.3Challenge-Generation Protocol Verification

This section discusses the application of the Scyther formal verification tool to the challenge-generation protocol after modelling and applying the adversary model. Based on the protocol description in section 5.5.1, the expected correct behaviour of the protocol is presented along with the actual verification results generated after applying the adversary model. Finally, from the infinite set of all traces, the finite possible attack patterns were manually investigated to be categorised into four patterns that can attack the protocol in the case of a compromised browser.

5.6.3.1 Adversary Modelling

Based on the adversary model presented in section 5.4, which has complete control over the browser through its ability to intercept and manipulate any information a user submits or receives in real time, therefore it is assumed that role $\boldsymbol{B}$ within the challenge-generation protocol is infected with the MITB attack.

As the MITB attack can bypass the encryption and decryption processes performed in the transport layer [1, 14]; therefore, the MITB can intercept and manipulate all exchanged cryptographic messages and their contents within the challenge-generation protocol, thus failing to satisfy the essential properties required by the proposed authentication system for both role.

For simulating the effect of the MITB attack within the browser $\boldsymbol{B}$ role, the adversary is modelled via revealing the shared secret key used for encrypting all exchanged messages within the challenge-generation protocol. In Scyther, this can be translated by revealing the actor's long-term key $\boldsymbol{K}(\boldsymbol{B},\boldsymbol{S})$, which represents the long-term symmetric key shared between the browser $\boldsymbol{B}$ and the server $\boldsymbol{S}$. Moreover, Scyther was set to find all possible attacks within both protocols.

5.6.3.2 Automated Challenge-Generation Verification

Although the proposed authentication system is built with a specific objective, there is still the slightest possibility of an honest agent representing the browser. Therefore, other possible attacks were considered, which urged the development of a secure protocol that employs cryptographic attributes to satisfy the security properties. Figure 5.6 and 5.7 shows the correct behaviour of the protocol execution in the case of an honest agent, with respect to both roles. As protocol execution is represented in terms of reachable states, both figures show all the states within role $\boldsymbol{S}$ and role $\boldsymbol{B}$ that are reached during the protocol execution. A proper description of figure 5.6 is provided below, which includes the assigned runs to each role, agent names running each role, fresh values and their corresponding variables. The same applies to figure 5.7, with variations in variable names.

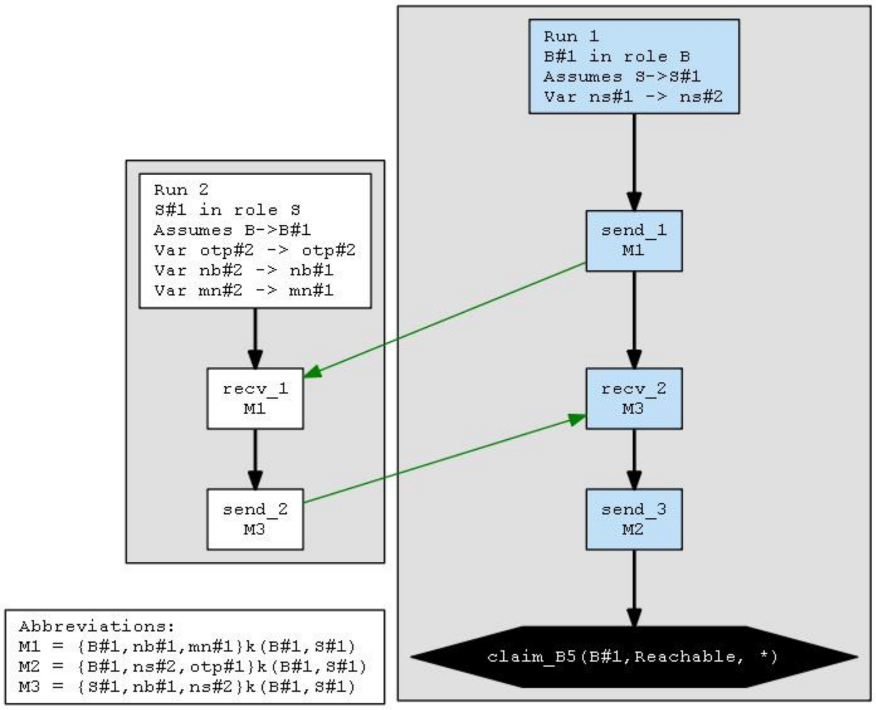


Figure 5.6: Correct Behavior of role B in the Challenge Generation Protocol

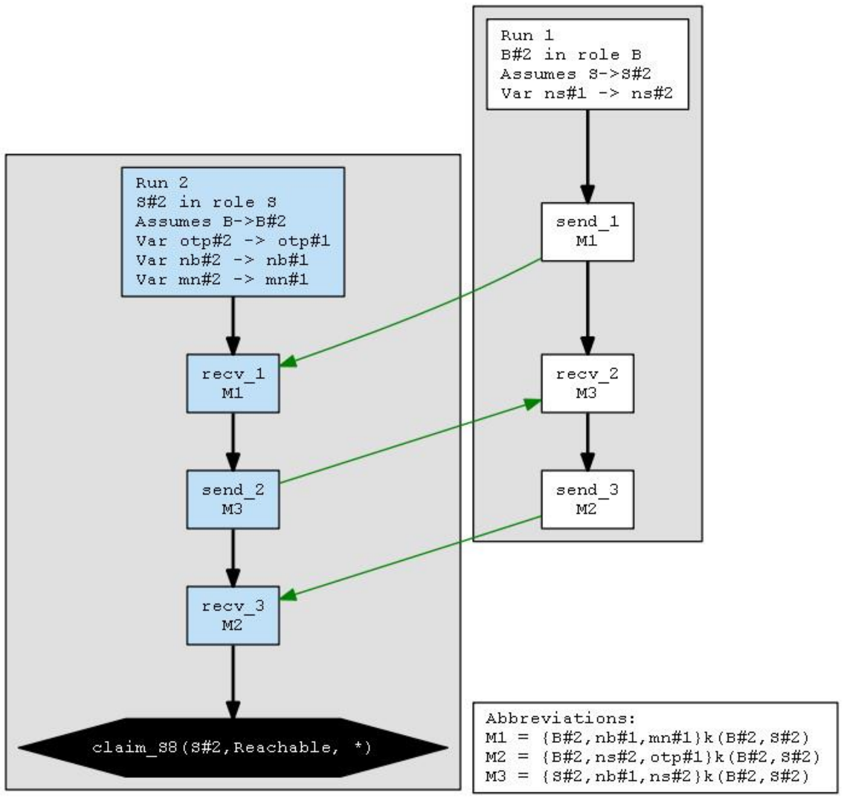


Figure 5.7: Correct Behavior of role S in the Challenge Generation Protocol

- The trace pattern in figure 5.7 shows that each instantiated role is assigned to the unique run identifier.

$$\boldsymbol{B} \longrightarrow \boldsymbol{Run1}$$

$$\boldsymbol{S} \longrightarrow \boldsymbol{Run2}$$

- Unlike the Message Sequence Chart (MSC) of the challenge generation protocol shown in figure 5.2, this figure shows that role names (B, S) are bind to the names of actual agents that are executing the roles ($B^{\#2}$, $S^{\#2}$). Thus, names of actual agents substitute role names in the exchanged messages within the trace pattern.

$$\{\boldsymbol{B} \longrightarrow \boldsymbol{B^{\#2}}, \boldsymbol{S} \longrightarrow \boldsymbol{S^{\#2}}\}$$

- Furthermore, fresh values are generated by both roles ($\boldsymbol{nb}, \boldsymbol{ns}$) are appended to the identifiers of their runs ($\boldsymbol{Run1}, \boldsymbol{Run2}$). The same applies to constant ($\boldsymbol{mn}$) and the ($\boldsymbol{otp}$) that is a defined user type.

$$\boldsymbol{Run1} \longrightarrow \boldsymbol{nb} \longrightarrow \boldsymbol{nb^{\#1}}$$

$$\boldsymbol{Run1} \longrightarrow \boldsymbol{mn} \longrightarrow \boldsymbol{mn^{\#1}}$$

$$\boldsymbol{Run1} \longrightarrow \boldsymbol{otp} \longrightarrow \boldsymbol{otp^{\#1}}$$

$$\boldsymbol{Run2} \longrightarrow \boldsymbol{ns} \longrightarrow \boldsymbol{ns^{\#2}}$$

As our protocol employs fresh values of type Nonce, agents running recipient roles use variables to store received terms. Upon sending these values, the fresh nonce $\boldsymbol{nb^{\#1}}$ generated by role $\boldsymbol{B}$ is stored in variable $\boldsymbol{nb^{\#2}}$ on the server side, and fresh nonce $\boldsymbol{ns^{\#2}}$ generated by role $\boldsymbol{S}$ is stored in variable $\boldsymbol{ns^{\#1}}$ on the browser side.

$$\textbf{Var } \boldsymbol{nb^{\#2}} \longrightarrow \boldsymbol{nb^{\#1}}$$

$$\textbf{Var } \boldsymbol{ns^{\#1}} \longrightarrow \boldsymbol{ns^{\#2}}$$

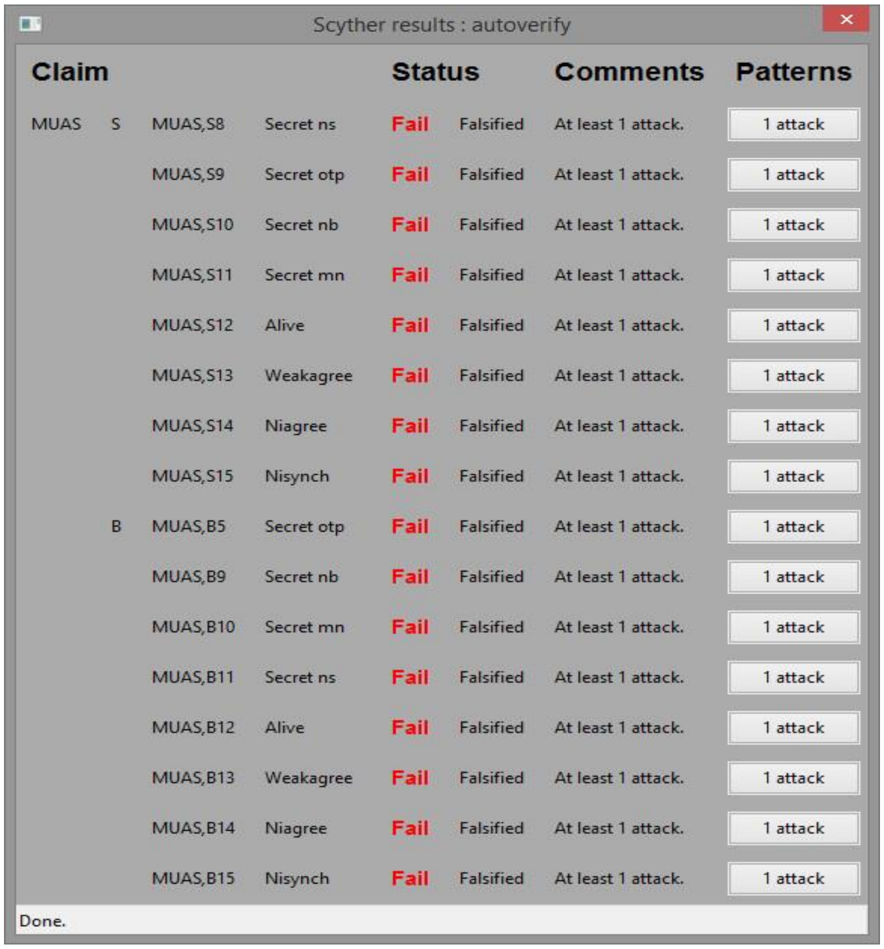
Similarly, upon sending the constant ($\boldsymbol{mn^{\#1}}$) and the user-defined ($\boldsymbol{otp^{\#1}}$) from $\boldsymbol{Run1}$ to $\boldsymbol{n2}$, the constant $\boldsymbol{mn^{\#1}}$ is stored in variable $\boldsymbol{mn^{\#2}}$, and the user-defined $\boldsymbol{otp^{\#1}}$ is stored in variable $\boldsymbol{otp^{\#2}}$.

$$\textbf{Var } \boldsymbol{mn^{\#1}} \longrightarrow \boldsymbol{mn^{\#2}}$$

$$\textbf{Var} \quad otp^{\#1} \longrightarrow otp^{\#2}$$

Upon the above configurations, the protocol starts exchanging the messages based on the protocol specifications.

On the other hand, applying the effects of the MITB attack on the protocol causes the Scyther's automatic claim verification process to scan a proof tree for all possible protocol behaviours to return the results of the verification process. The size of the bounded proof tree ensures the verification process terminates and returns the results before reaching the upper bound [257]. In our case, the challenge-generation protocol was verified for bounded and unbounded verification, where the bounded verification was set to the default value of the tool, namely 5. However, verification was reached and the result was generated as described in Figure 5.8.



Scyther results : autoverify

Claim				Status		Comments	Patterns
MUAS	S	MUAS,S8	Secret ns	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,S9	Secret otp	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,S10	Secret nb	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,S11	Secret mn	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,S12	Alive	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,S13	Weakagree	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,S14	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,S15	Nisynch	Fail	Falsified	At least 1 attack.	1 attack
	B	MUAS,B5	Secret otp	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,B9	Secret nb	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,B10	Secret mn	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,B11	Secret ns	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,B12	Alive	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,B13	Weakagree	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,B14	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		MUAS,B15	Nisynch	Fail	Falsified	At least 1 attack.	1 attack

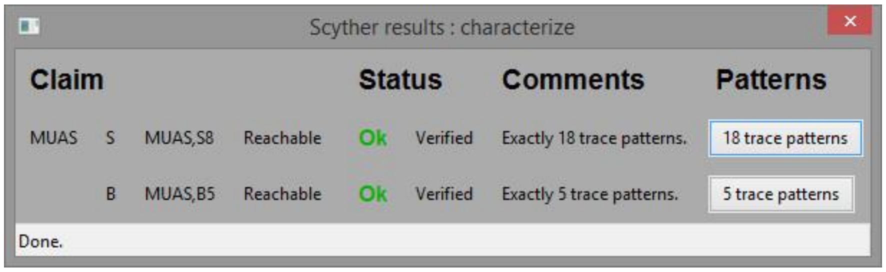
Done.

Figure 5.8: Challenge-generation protocol verification results

Figure 5.8 shows the result window of both roles within the protocol, in which the secrecy of the freshly generated numbers by browser $\boldsymbol{B}$ and server $\boldsymbol{S}$, denoted by $\boldsymbol{nb}$ and $\boldsymbol{ns}$, respectively, failed to be satisfied in both roles. Moreover, both roles of the challenge-generation protocol failed to satisfy all forms of the authentication property, namely: aliveness, weak-agree, non-injective agreement and non-injective synchronisation.

5.6.3.3 Automated Challenge-Generation Analysis

Upon simulating the MITB effects and setting bounded or unbounded verification parameters, the protocol is analysed through a complete characterisation of each role. Thus, for each role, a finite pattern representation is efficiently generated from an infinite set of traces that contain an instance of that role [253, 256, 258]. The complete characterisation of the challenge-generation protocol is shown in Figure 5.9.



Scyther results : characterize

Claim				Status		Comments	Patterns
MUAS	S	MUAS,S8	Reachable	Ok	Verified	Exactly 18 trace patterns.	18 trace patterns
	B	MUAS,B5	Reachable	Ok	Verified	Exactly 5 trace patterns.	5 trace patterns

Done.

Figure 5.9: Complete characterization of the challenge-generation protocol

By manually reviewing these patterns, it was found that for each role, one pattern represents the correct behaviour as seen in figure 5.6 and 5.7, while the rest of the patterns represent all possible security attacks on the challenge generation protocol. The representative attack patterns show that: (a) the protocol can suffer from attackers that compromise exchanged messages, (b) an adversary can run the protocol via impersonating the communicating partner, (c) multiple instances of different roles are running the same protocol via an adversary and (d) multiprotocol attacks can occur as well. An example of each possible attack graph is shown in Figures 5.10(a), 5.11(b), 5.12(c) and 5.13(d).

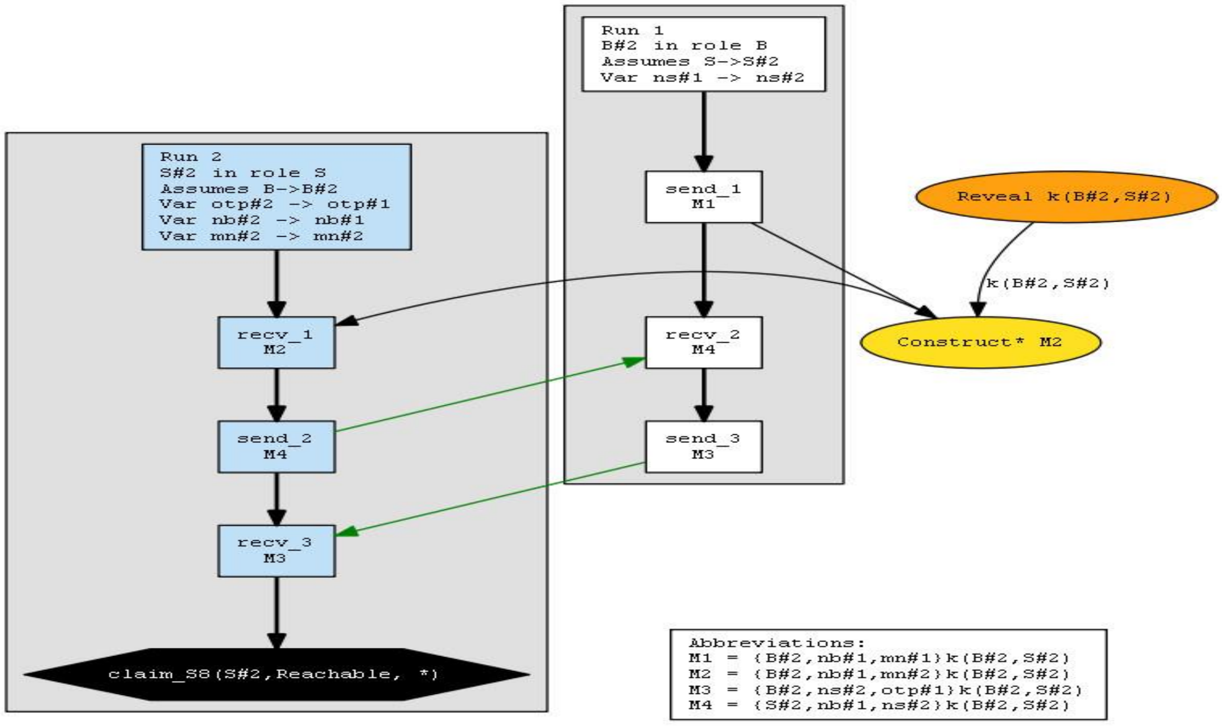


Figure 5.10: An attack graph of the challenge-generation protocol (a)

Figure 5.10 shows an adversary intercepting exchanged messages, and manipulating their contents. In this figure, the adversary compromised $\boldsymbol{M1}$ that is sent by browser $\boldsymbol{B}$, and replaced the submitted mobile number $\boldsymbol{mn}$ by another number before submitting the message to server $\boldsymbol{S}$. This prevents the legitimate client from receiving the SMS-OTP, hence disabling the protocol.

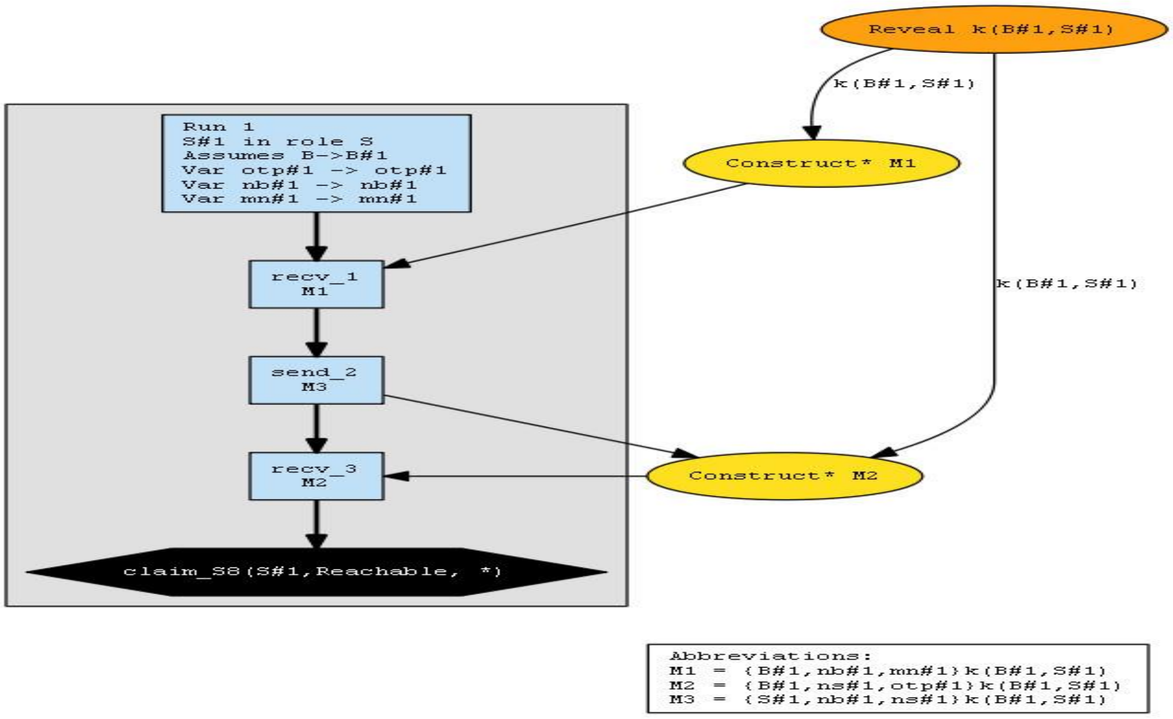


Figure 5.11: An attack graph of the challenge-generation protocol (b)

Upon session key reveal, the adversary can intercept and decrypt exchanged messages. Thus, the identity of the sender can be compromised as all exchanged messages include the client's identity. Figure 5.11 shows the adversary in the process of impersonating role B and communicating with server S. The adversary utilises the browser's compromised identity to construct M1 and sends it to the server, thus causing server S to believe that he is communicating with the legitimate client B.

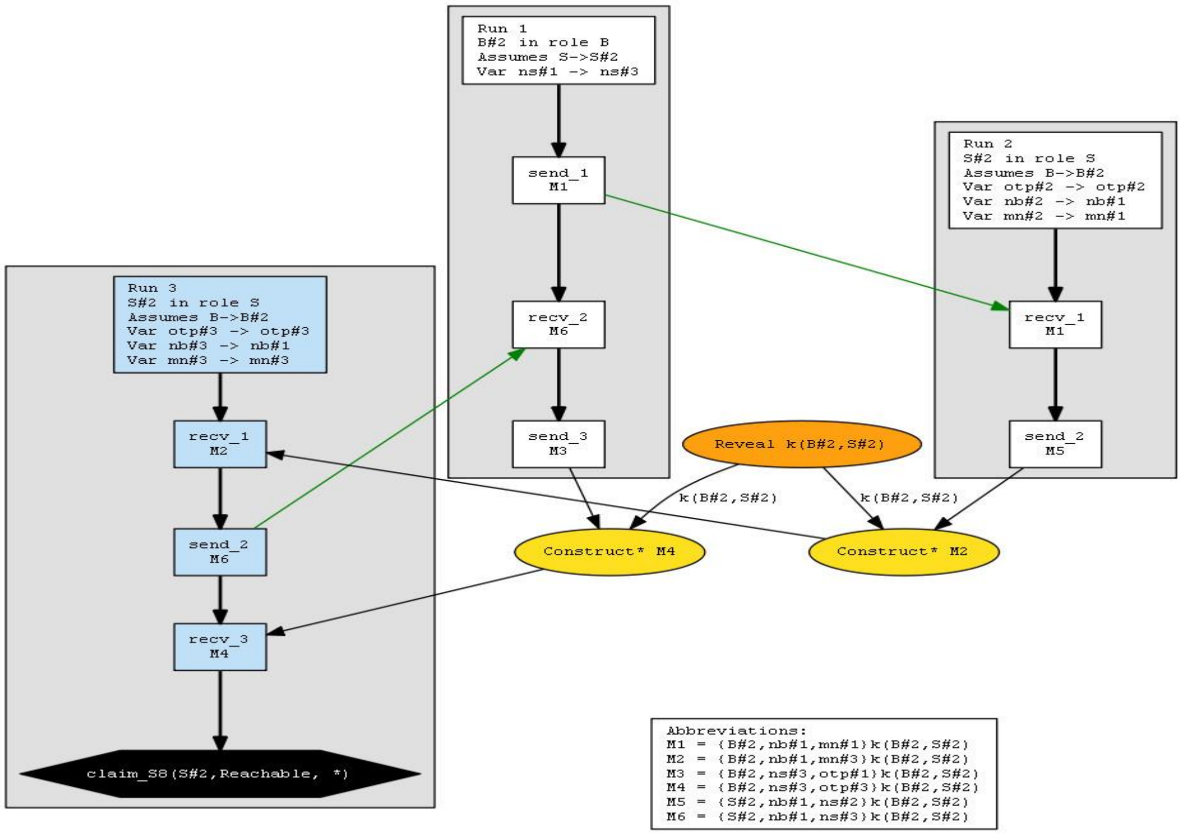


Figure 5.12: An attack graph of the challenge-generation protocol (c)

As agents can execute any number of runs in parallel, in an interleaved manner. Figure 5.12 shows that agent $\boldsymbol{S}^{\#2}$ is executing the same $\boldsymbol{S}$ role in two different runs ($\boldsymbol{Run2}, \boldsymbol{Run3}$). In this figure, $\boldsymbol{Run1}$ ($\boldsymbol{role\ B}$) initiates a connection with $\boldsymbol{Run2}$ ($\boldsymbol{role\ S}$), whose response to $\boldsymbol{Run1}$ ($\boldsymbol{role\ B}$) has been intercepted and manipulated by an adversary that utilises the browser nonce $\boldsymbol{nb}^{\#1}$ for impersonating a browser and trying to initiate a new connection with $\boldsymbol{Run3}$ ($\boldsymbol{role\ S}$). Thus, the second protocol execution occurs between $\boldsymbol{Run1}$ ($\boldsymbol{role\ B}$) and $\boldsymbol{Run3}$ ($\boldsymbol{role\ S}$). Hence, $\boldsymbol{Run3}$ ($\boldsymbol{role\ S}$) responds back with the browser nonce $\boldsymbol{nb}^{\#1}$ and a new server nonce $\boldsymbol{ns}^{\#3}$ to $\boldsymbol{Run1}$ ($\boldsymbol{role\ B}$), assuming that the received response is from $\boldsymbol{Run2}$ ($\boldsymbol{role\ S}$). Upon receiving the response and verifying the browser nonce $\boldsymbol{nb}^{\#1}$, the response of $\boldsymbol{Run1}$ ($\boldsymbol{role\ B}$) is intercepted and manipulated by the adversary to get the server nonce $\boldsymbol{ns}^{\#3}$**.** Finally, the adversary sends the manipulated response to $\boldsymbol{Run3}$ ($\boldsymbol{role\ S}$) instead of $\boldsymbol{Run2}$ ($\boldsymbol{role\ S}$). Hence, $\boldsymbol{Run1}$ ($\boldsymbol{role\ B}$) is running the protocol with two different runs ($\boldsymbol{Run2}$ ($\boldsymbol{role\ S}$), $\boldsymbol{Run3}$ ($\boldsymbol{role\ S}$)).

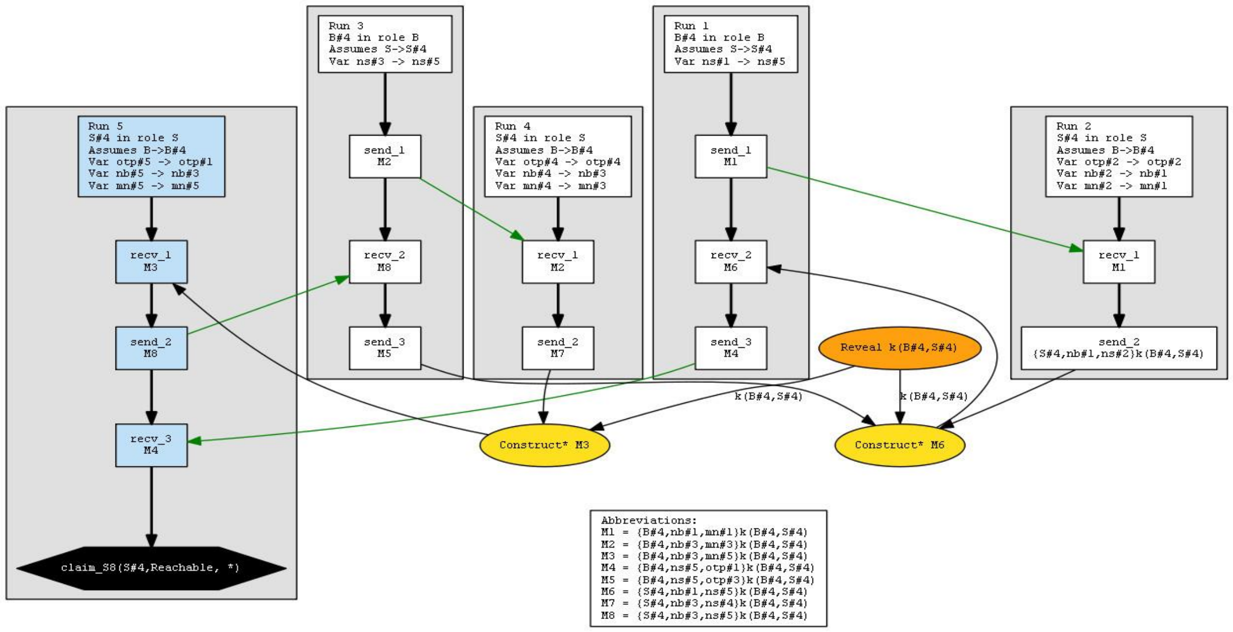


Figure 5.13: An attack graph of the challenge-generation protocol (d)

Similar to the previous figure, figure 5.13 shows that agent $\boldsymbol{S^{\#4}}$ is executing the same $\boldsymbol{S}$ role in three different runs ($\boldsymbol{Run2}, \boldsymbol{Run4}, \boldsymbol{Run5}$), and that agent $\boldsymbol{B^{\#4}}$ is executing the same $\boldsymbol{B}$ role in two different runs ($\boldsymbol{Run1}, \boldsymbol{Run3}$). In this figure, the first protocol execution occurs between $\boldsymbol{Run3}$ ($\boldsymbol{role\ B}$) and $\boldsymbol{Run4}$ ($\boldsymbol{role\ S}$), whose response to $\boldsymbol{Run3}$ ($\boldsymbol{role\ B}$) has been intercepted and manipulated by an adversary that utilizes the browser nonce $\boldsymbol{nb^{\#3}}$ for impersonating a browser and trying to initiate a new connection with $\boldsymbol{Run5}$ ($\boldsymbol{role\ S}$). Thus, the second protocol execution occurs between $\boldsymbol{Run3}$ ($\boldsymbol{role\ B}$) and $\boldsymbol{Run5}$ ($\boldsymbol{role\ S}$). Hence, $\boldsymbol{Run5}$ ($\boldsymbol{role\ S}$) responds back with the browser nonce $\boldsymbol{nb^{\#3}}$ and a new server nonce $\boldsymbol{ns^{\#5}}$ to $\boldsymbol{Run3}$ ($\boldsymbol{role\ B}$), assuming that the received response is from $\boldsymbol{Run4}$ ($\boldsymbol{role\ S}$). However, upon receiving the response and verifying the browser nonce $\boldsymbol{nb^{\#3}}$, the response of $\boldsymbol{Run3}$ ($\boldsymbol{role\ B}$) is intercepted and manipulated by the adversary to get and include the server nonce $\boldsymbol{ns^{\#5}}$ in a third protocol execution that occurs between $\boldsymbol{Run1}$ ($\boldsymbol{role\ B}$) and $\boldsymbol{Run2}$ ($\boldsymbol{role\ S}$). In this protocol, $\boldsymbol{Run1}$ ($\boldsymbol{role\ B}$) initiates a connection with $\boldsymbol{Run2}$ ($\boldsymbol{role\ S}$), whose response to $\boldsymbol{Run1}$ ($\boldsymbol{role\ B}$) has been intercepted and manipulated by the adversary that altered the server nonce from $\boldsymbol{ns^{\#2}}$ to $\boldsymbol{ns^{\#5}}$ before being sent to $\boldsymbol{Run1}$ ($\boldsymbol{role\ B}$). Finally, $\boldsymbol{Run1}$ ($\boldsymbol{role\ B}$) sends the final response to $\boldsymbol{Run5}$ ($\boldsymbol{role\ s}$), assuming that the received response is from $\boldsymbol{Run3}$ $(\boldsymbol{role\ B})$. Thus, this trace pattern is a multiprotocol attack that is comprised of two browsers; each is running the protocol with two different runs.

Based on the generated attack patterns, the consequences of the MITB attack compromising the browser cannot be circumvented within the same protocol. As a strong intruder compromising an agent and taking control of the entire browser can corrupt random number generators and reveal long-term and short-term keys [241]. Hence, there is no way to prevent this type of attack along with overcoming a denial of service (DOS) attack, unless transitioning to another communication channel with a second protocol.

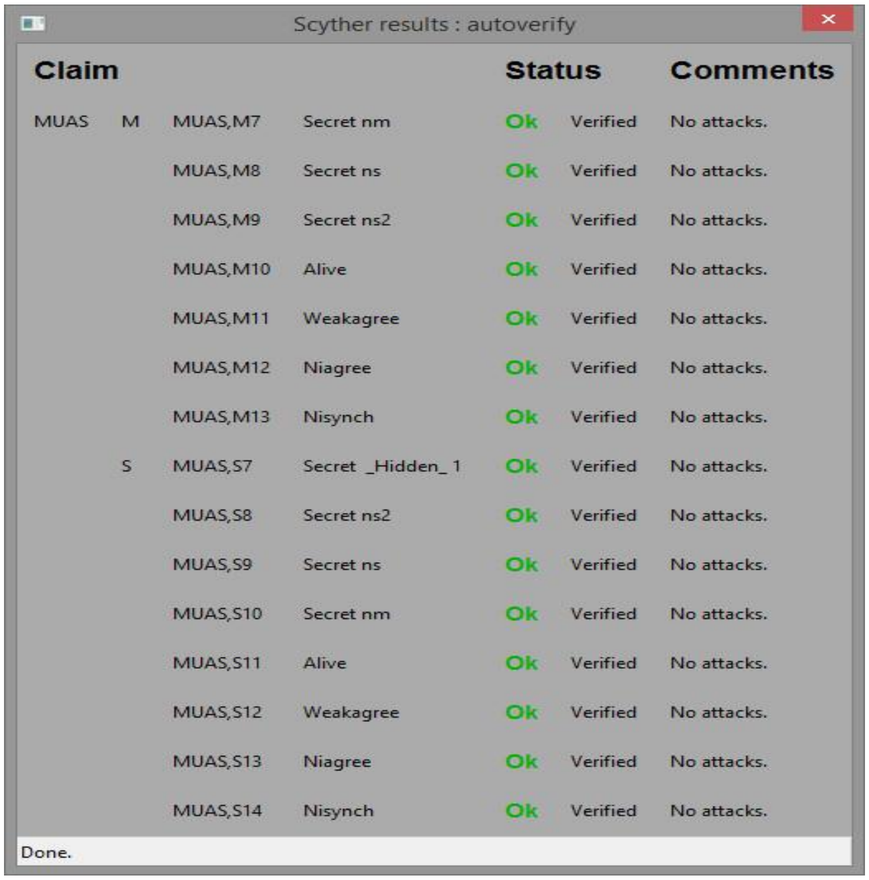
5.6.4Response-Generation Protocol Verification

Following the unsuccessful verification of the challenge-generation protocol, for the MITB attack compromising the browser, and revealing its long-term shared symmetric key, a complementary and detached protocol is established for generating the response with no risk of the same adversary.

This section discusses the application of the Scyther formal verification tool on the response-generation protocol. Based on the protocol description in section 5.5.2, the expected correct behaviour of the protocol is presented along with its actual verification results. The execution of claim events that specify the required security properties are justified and analysed, along with presenting all possible trace patterns of both roles, generated via complete characterisation.

5.6.4.1 Automated Response-Generation Verification

The response-generation protocol is automatically and separately verified through setting bounded and unbounded verification parameters. Thus, the protocol verification procedure terminates and the results are generated, as illustrated in Figure 5.14, where the result window of both roles within the response-generation protocol, in which the secrecy of the freshly-generated numbers by mobile phone $\boldsymbol{M}$ and server $\boldsymbol{S}$, denoted by $\boldsymbol{nm}$ and $\boldsymbol{ns2}$, respectively, are successfully verified in both roles. In addition, the response-generation protocol successfully verified the *Aliveness, weak-agree, non-injective agreement* and *non-injective synchronisation* claim events, which will be analysed in-depth in the next section.



Scyther results : autoverify

Claim				Status		Comments
MUAS	M	MUAS,M7	Secret nm	Ok	Verified	No attacks.
		MUAS,M8	Secret ns	Ok	Verified	No attacks.
		MUAS,M9	Secret ns2	Ok	Verified	No attacks.
		MUAS,M10	Alive	Ok	Verified	No attacks.
		MUAS,M11	Weakagree	Ok	Verified	No attacks.
		MUAS,M12	Niagree	Ok	Verified	No attacks.
		MUAS,M13	Nisynch	Ok	Verified	No attacks.
	S	MUAS,S7	Secret _Hidden_ 1	Ok	Verified	No attacks.
		MUAS,S8	Secret ns2	Ok	Verified	No attacks.
		MUAS,S9	Secret ns	Ok	Verified	No attacks.
		MUAS,S10	Secret nm	Ok	Verified	No attacks.
		MUAS,S11	Alive	Ok	Verified	No attacks.
		MUAS,S12	Weakagree	Ok	Verified	No attacks.
		MUAS,S13	Niagree	Ok	Verified	No attacks.
		MUAS,S14	Nisynch	Ok	Verified	No attacks.

Done.

Figure 5.14: Response-generation protocol verification results

5.6.4.2 Analysis of Response-Generation Security Properties

In this section, the automated verified properties from section 5.6.4.1 are discussed and analysed. These properties are analysed according to the design of the proposed response generation protocol. The verified properties are secrecy and the different forms of authentication, which are aliveness, synchronisation and message agreement.

.

5.6.4.2.1 Secrecy/Confidentiality

According to the description of the response-generation protocol in section 5.5.2, the secrecy property of all essential elements within the protocol, including nonce, macros, variables and constants, are satisfied. Secrecy satisfaction lies within securing the protocol via symmetrical encryption of exchanged messages between two honest agents, in which their identities have been verified via PKI.

5.6.4.2.2 Authentication

a) Aliveness

In terms of verifying the aliveness property, all forms of aliveness are analysed regarding both roles in the response-generation protocol. Moreover, both partners within the protocol are considered honest due to their shared secret key, which was established previously within the SSL/TLS protocol. As both agents within the protocol execute both roles up to their claim events, and both intended communicating partners are *honest* due to an event execution. Thus, response-generation protocol assures agents execute the *weak aliveness* role of their intended communicating partners. The *weak aliveness in correct role* property is satisfied within the response-generation protocol as both parties act in their correct roles due to their shared secret keys used to symmetrically encrypt all exchanged messages between communicating parties, which ensures integrity and confidentiality as well. Thus, the intended communicating partners are executing the roles expected from the protocol description [41].

As both agents within the protocol execute their roles up to their claim events, thus ensuring that the intended communicating partners has recently sent messages before the claim events and after other events within the same run in which the claim event occurs, which ensures that *recent aliveness* property is satisfied. Finally, based on the shared secret key between both parties in the protocol, intended communicating partners execute the correct roles, thus satisfying the *recent aliveness in correct role* property. Consequentially, the *weak agreement* property is satisfied due to the established shared secret key between both participants [241]. Based on figure 5.4, any protocol satisfying *recent aliveness* also satisfies *weak aliveness* [41]. Thus, the response-generation protocol satisfies *weak aliveness*, *weak aliveness in the correct role*, *recent aliveness* and *recent aliveness in the correct role*.

a) Synchronisation

According to the definition of the synchronisation property in section 5.6.1.2.2, the response generation protocol satisfies synchronization. Figures 5.15 and 5.16 show the Scyther trace patterns of both roles within the response-generation protocol that corresponds to the protocol description in section 5.5.2.

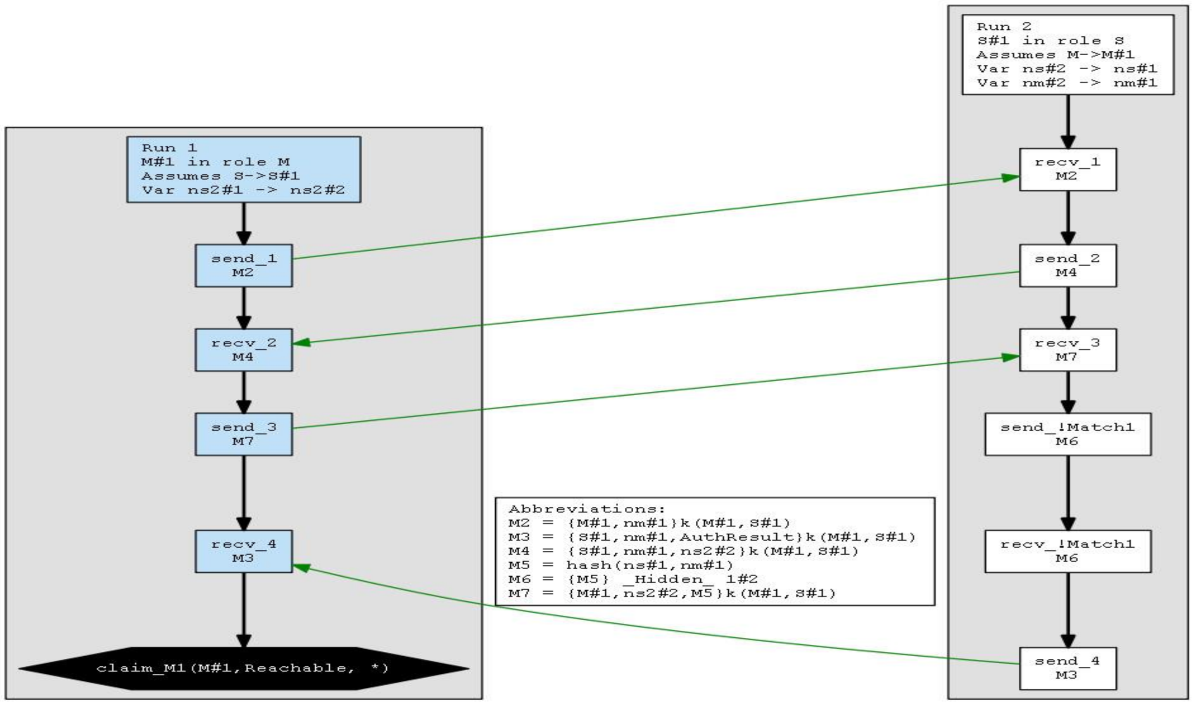


Figure 5.15: Trace pattern of the response-generation protocol, claim MUAS, M1 in role M

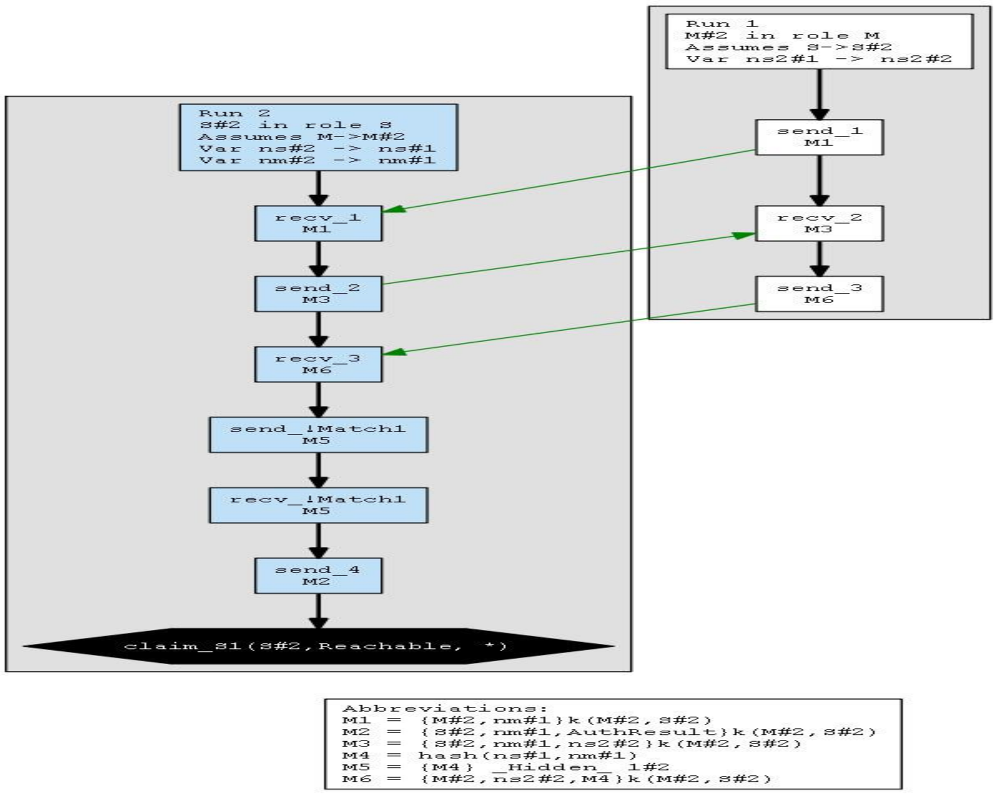


Figure 5.16: Trace pattern of the response-generation protocol, claim MUAS, S1 in role S

Figures 5.15 and 5.16 confirm the non-injective synchronisation property for both roles in the protocol, where the roles are fulfilled through assigning each claim instance and role to a run identifier through cast functions. With respect to the characterised role, both roles in the protocol are assigned to Run1 and Run2.

Once the communication partners are assigned, it is required for all communication pairs preceding the claim event to: (1) occur in the correct order, where every received message is preceded by a corresponding send message, (2) trace events corresponding to the events in the protocol description, and executed by the runs indicated by the cast function, which are Run1 and Run2, and (3) have the contents of the send messages correspond to the contents of the received messages [41, 247, 252]. According to the trace pattern results, both protocols satisfy the non-injective synchronisation property with respect to both roles, thus ensuring the protocol is executed as expected.

However, non-injective synchronisation does not rule out replay attacks, where an intruder is capable of abusing the protocol by replaying an old and unexpected responder run into the current session. To rule out such a flaw, an *injectivity* property is added to the protocol, which requires that each run of the initiator role has his own and unique nonce to be included in all exchanged messages with its corresponding responder role. This injective one-one relationship between the initiator role and the responder role causes the responder to only correspond to a single initiator role, thus creating a loop from the initiator to the responder and back to the initiator, which prevents replay attacks [41, 247, 249, 251, 252].

Based on Figure 5.15, any protocol satisfying injective synchronisation also satisfies non-injective synchronisation [41]. Thus, the challenge- and response-generation protocols satisfy both forms of synchronisation.

b) Message Agreement/Integrity

Based on the trace events in Figures 5.15 and 5.16, the non-injective agreement property is confirmed, where both roles in the protocols agree on the contents of the corresponding send and receive messages. Moreover, according to the synchronisation hierarchy in Figure 5.5, a protocol satisfying injective synchronisation also satisfies non-injective synchronisation, thus satisfying the non-injective agreement.

5.7 Conclusion

This chapter complements the design and implementation of the proposed MUAS system in a formal means through verifying the proposed protocol from a security point of view. Therefore, a theoretical security evaluation has been performed with respect to possible attacks for justifying every specific detail within the protocol design. In addition to evaluating the implementation performance with respect to the most related work, where the evaluation is conducted in terms of communication and computation costs. The evaluation results showed our protocol has the most messages in terms of quantity and size. However, a trade-off has been made with respect to security. Moreover, upon conducting a run-time measurement on both protocols, it was found that the response generation protocol is much faster than the input-based challenge generation protocol.

For performing a proper verification mechanism, the objective behind the development of the protocol should be defined. Hence, the adversary model has been defined within the context of the proposed protocol. The adversary model in this thesis is represented by the

MITB attack, which resides in the browser; thus, the adversary can intercept and manipulate any data entered, received or sent through the compromised browser. The next building block in the verification mechanism lies within specifying the protocol in an abstract manner to declare the required knowledge for role execution within the protocol. The protocol specification is then translated into the selected verification tool's description language.

Among numerous formal model checking tools, the Scyther security protocol verification tool was considered to verify the required security properties of the protocol due to its speed and other novelties [239, 241, 253, 254, 259]. Based on chapter 3, the MUAS approach split the authentication mechanism into two detached protocols: challenge-generation and response-generation protocols. Thus, both protocols are verified separately using Scyther.

The first protocol was verified upon modelling and applying the effects of the MITB attack within the protocol. In Scyther, the MITB attack was modelled by revealing the actor's long-term key, which corresponds to the established shared secret key used for securing all exchanged messages between the server and the compromised browser within the challenge-generation protocol. This type of exposure was based upon the MITB's capability of bypassing the encryption and decryption processes performed in the transport layer [1, 14]. Upon applying the adversary model, setting unbounded verification parameters and performing an automated verification, a proof tree is scanned for all possible protocol behaviours to return the results of the verification process. The result of verifying the challenge-generation protocol has failed for all required security properties. Moreover, a complete characterisation for each role is performed, thus generating a finite set of all possible traces that contains an instance of that role. These traces include the anticipated and correct behaviour, in addition to all possible attack patterns that have been manually reviewed and categorised into four classes, namely, exchanged messages manipulation, communicating partner impersonation, running the same protocol through multiple instances of the same role, and multiprotocol attacks. Based on the generated attack patterns, the MITB attack cannot be prevented within the same protocol as it took over the whole system. Thus, authentication via web browser is not secure. The proposed solution is based upon establishing a physically detached protocol that circumvents the MITB attack through transitioning the authentication mechanism to another communication channel that uses the mobile phone of the client. This protocol generates a response to the challenge with respect to authenticating the client, along with its ability to conduct a secure online transaction.

The second detached protocol takes place between the mobile phone application '3LS-Authentication' and the server, which is initiated following the challenge capturing via the visual channel. As the mobile phone application is secured by an activation layer, its role is considered to be executed by an honest agent without the need for modelling an adversary; therefore, all protocol communications are secured via their shared secret key. However, upon setting the unbounded verification parameters within the verification process by Scyther, secrecy and all forms of authentication security properties were satisfied. Furthermore, each role has exactly one trace pattern generated by the complete characterisation process, which corresponds to a valid protocol execution with no attacks [255, 256].

6 Conclusion and Future Work

6.1 Introduction

E-commerce applications have steadily grown in recent years. This growth is accompanied by many security concerns that have become obstacles to the development of e-commerce. User authentication is considered one of the most important issues to be addressed within the context of e-commerce; it is the first step in establishing a secure connection with any organisation. Therefore, the focus of this thesis revolves around finding an appropriate solution that overcomes the problem of a Man-in-the-browser (MITB) attack, within a secure e-commerce authentication system. An MITB attack is a Trojan that invades a user's browser to infiltrate and manipulate all incoming and outgoing information. This type of attack is a sophisticated hacking technique [11] capable of bypassing the security measures of Public key infrastructures (PKIs) and the encryption mechanisms of a Secure socket layer and Transport layer security (SSL/TLS) [1, 14].

In this chapter, we provide a summary of the stages involved in developing the proposed Mobile user authentication system (MUAS) along with its learning outcomes. Beginning with the literature review, we investigated many articles to guide the design and implementation of the proposed system that was ultimately evaluated and verified as a high performance solution that prevented MITB attacks.

In conclusion, the main goal of this thesis is to develop a secure Mobile-User-Authentication System (MUAS), for circumventing MITB attacks through switching from the infected platform to the mobile phone application. Hence, the mobile phone is used for authentication and for conducting business transactions.

6.2 Summary of the Results

The focus of this thesis is to design and implement a two-phase MUAS that overcomes an MITB attack. The design was evaluated using a reliable verification method that authenticated the properties of the proposed security protocol. The learning outcomes of this thesis are summarised in the list below:

- In chapter 1, we discussed the sophisticated MITB attacks. We analysed possible attack capabilities and performance behaviour in order to find a way that circumvent its destructive effects. The objectives and research contributions are outlined in this chapter as well.

- In chapter 2, we delineated the development stages of the system design. First, we reviewed the three fundamental authentication methodologies and conducted a comparative study of security protocols employed by popular UK banks. Results of the study revealed that the most frequently used identification and authentication mechanism is 2FA, due to its secure login process. However, 2FA cannot accomplish the objective of this thesis, because it, uses the same mechanisms that the infected browser uses to fulfil the authentication process. Instead, a mobile phone, which is the most popular platform associated with 2FA mechanisms, is employed in the proposed thesis. A mobile phone has many pre-existing security-feature options available, often stemming from popular technologies. Next, we reviewed the e-commerce security system. The e-commerce security system represents the foundation of our work. The system is composed of five basic layers: a *network service layer*, *encryption technology layer*, *security authentication layer*, *security protocol layer* and *application layer*. Each layer serves a specific purpose within the security system. The lower-most layer is the *network service layer*, which represents the data link layer of the e-commerce security system. From a security protocol point-of-view, the next three layers represent the encryption technology layer, security authentication layer and security protocol layer. Each of the three layers represents essential and basic elements for developing a secure e-commerce authentication system, which protects personal information from unauthorized use while satisfying the required security dimensions. From a security dimension point-of-view, encryption technology layers help satisfy the integrity, non-repudiation, authentication and confidentiality dimensions [24]. Security authentication layers help satisfy integrity, authenticity and non-repudiation dimensions [24], while security protocol layers help satisfy the authenticity, integrity, confidentiality and privacy dimensions [189-191]. Therefore, this system is considered a platform that determines the technologies that need to be implemented within each layer in the proposed authentication system.

- The *encryption technology layer* secures transmitted data from unauthorized use via cryptographic algorithms. Thus, the encryption layer is considered a fundamental layer in the field of e-commerce security [154]. There are two encryption categories: asymmetric and symmetric cryptography. Asymmetric cryptography provides several security mechanisms such as key establishment, non-repudiation, and entity identification. Although symmetric cryptography can identify entities, identifying key establishment requirements is significantly more difficult [160] because it poorly satisfies the non-repudiation requirement [157, 160]. Moreover, asymmetric cryptography is very slow, compared to symmetric cryptography. The system design must have an explicit symmetric or asymmetric cryptography type to ensure the rationality/logic of the key distribution and encryption efficiency [20]. In some cases, choosing an explicit asymmetry type is a trade-off with respect to the objective of the proposed system.
- The *secure authentication layer* of an e-commerce system represents the basis for authenticating the identities of participants and their exchanged messages. This can be accomplished by establishing a robust infrastructure that provides e-commerce security services. Therefore, a public key infrastructure (PKI) represents the security foundation of MUAS as it can provide the infrastructure for various network security services. Additionally, PKIs can provide different public key certificate services [22],

such as obtaining public keys that can be used in the encryption technology layer for cryptographic purposes. Moreover, other security technologies, such as cryptographic nonce, can be used to ensure the authenticity of exchanged messages.

  - The *security protocol layer* is a necessary way of establishing a secure channel between participants like e-commerce or online banking. The typical choice is usually the secure socket layer/transport layer security (SSL/TLS) protocol. This protocol uses the client’s certificate as well as the CA-certificate within a PKI. The protocol then verifies the user’s identity and establishes a safe and trustworthy channel, between the communicating parties, to prevent transmitted data from being stolen [20].
  - The *application layer* represents the point of compromise. In this thesis, the threat model lies within this final layer of the e-commerce security system.

- In chapter 3, the details of the proposed system are provided, along with a justification of the different components and functions of the design. Based on the literature review and the thesis objective, the most appropriate solution is to securely transition the authentication process from an MITB-infected browser to the mobile phone of the client. This is accomplished by designing two-detached secure phases, called the challenge- and response-generation phases, each using distinct network connections. Based on the MUAS design architecture, several outcomes have been determined:
  - The importance of registering the client via a mobile number and the IMEI lies within mitigating denial-of-service attacks in both phases, where unregistered clients will be denied in the challenge generation phase. In the response generation phase, submitting the extracted mobile number and IMEI to the server mitigates the DOS attack as well, typically by matching the received mobile number with the registered client’s mobile number, to find and validate the received IMEI against the registered IMEI.
  - Whenever a registered client triggers the authentication system, the client’s registered mobile phone is the only device that can generate the appropriate response for that session. The IMEI is extracted from the mobile phone while generating the response that will be sent to the server and verified against the registered IMEI; hence, it is not possible for multiple simultaneous (or cloned) mobile phones to run the same system.

This protection is an additional way to guarantee the registration of the mobile phone, which sends the response.

- With the expansion of wireless access technologies such as Wi-Fi, Wi-Max, and LTE, and with the popularity of multi-interfaced mobile devices, efficient multi-homing support is possible in mobile networks. In a multi-homed mobile network, the mobile phone is considered the multi-homed mobile host, since its Mobile IP address changes while the selected network interface changes. Interface selection mechanisms in mobile networks are located in the mobile router (MR) [260]. The mobile router is responsible for updating its attachment point to the mobile network without disrupting higher-layer connections of attached devices, thus the mobile router oversees all the mobility functions on behalf of all the mobile hosts located within the mobile network. To find the best approach that supports network mobility and multi-homing, a study was conducted among several multi-homed mobile networks that resulted in the Network Mobility (NEMO) protocol, which provides support to both network mobility and multi-homing, in addition to providing better interface selection mechanism than other approaches [260]. NEMO's basic support protocol is the Mobile IPv6, which has been extended to support connectivity within a moving network [260, 261]. A mobile network contains one or more mobile routers (MRs), along with local or transient nodes [260]. A multi-homed response generation protocol enables the client to use a mobile phone and server as a mobile host (MH) and a correspondent node (CN), respectively. Regardless of the physical location of the mobile phone, it is identified by a permanent home address that is assigned by its home network. The server (CN) uses the permanent home address of the mobile phone as the destination address. Whenever the mobile phone moves to another network (foreign network), it acquires a new address called Care-of-Address (CoA). The mobile phone then sends its CoA to its home agent (HA), which is a router in a static location on the MHs home network that is responsible for tunnelling packets for delivery to the mobile phone when it is away from its home network. Next, a binding takes place between the mobile phone's permanent home address and the CoA. Thus, a bi-directional IPv6-in-IPv6 tunnel is used to maintain connectivity between the mobile router and the home agent. Response generation exchange message-packets sent from the server (CN) to the mobile phone on a mobile network

(MN) are delivered to the home agent of that mobile network via traditional routing methods. The home agent tunnels the packets that will delivered to the mobile phone. Likewise, reverse packets take the same path in the opposite direction. The mobile phone sends the packets to the mobile router, which will then be tunnelled to the home agent and finally sent to the server (CN) using traditional routing methods.

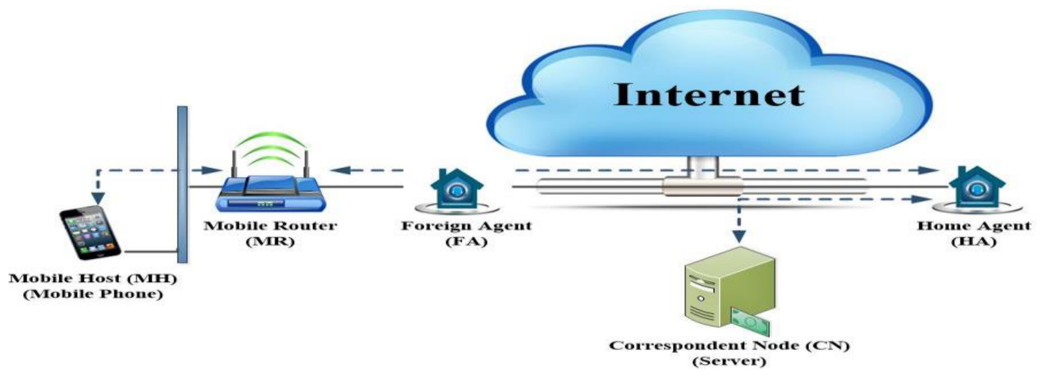

Internet
Mobile Router (MR)
Foreign Agent (FA)
Home Agent (HA)
Mobile Host (MH) (Mobile Phone)
Correspondent Node (CN) (Server)

Figure 6.1: Response Generation Protocol in a Multi-homed mobile Network

- In chapter 4, the theoretical design for the MUAS is interpreted as an executable form; the form defines the design components and provides descriptive flowcharts. This chapter represents the foundation on which the verification chapter is built-upon. The employed functions and cryptographic algorithms that are required for calculating the computational cost of the protocol are described in this chapter. Messages exchanged within the protocol are specified as well, along with their attributes for calculating the communication cost of the protocol. Furthermore, implementing the proposed system helps by calculating the average run-time of each phase within the protocol.

- In chapter 5, both phases of the proposed system are evaluated from a theoretical and applied point-of-view. The process also generates the verification results by applying formal verification tools.
    - An in-depth justification of all the exchanged messages, their contents and their applied cryptographic mechanisms contribute to a theoretical performance measure. This measurement justifies the system's security against all possible attacks. In the literature review, most related work [130, 200, 201] demonstrated an implementation performance measure that fulfils the design choices from the point-of-view of the computational and communication-based cost. The MUAS computational cost is higher in terms

of random number generation, while its energy consumption is lower since it employs a simple and lightweight algorithm. A public key cryptography method is employed for securing the QR-code content, and though it is expensive, a trade-off was made between the key cryptography and QR-code content. The MUAS communication cost exceeds that of other protocols in terms of the number of exchanged messages and the message sizes. The system is composed of two detached secure protocols and each is composed of several messages that are supported by the cryptographic security attribute and the sender’s identity.

- Each MUAS phase is evaluated separately, using the Scyther security protocol verification tool. The evaluation of the challenge-generation phase resulted in a failure; this failure applies to all required security properties. The failure is a consequence of applying the MITB effects on the protocol. The results also suggest that MITB attacks cannot be prevented within the same (single) protocol, especially when the attacker has overtaken the entire system. This has justified the need for relocating the authentication mechanism to a physically detached phase, which employs a secure authentication device (e.g., a mobile phone) that communicates with the server over a newly created connection. Evaluating the detached response generation phase results for fulfilling the confidentially (secrecy) and authentication security properties. The evaluations must validate the entire two-phase-detached protocol. Hence, a detached-phase approach of the security measure satisfies the objective of the thesis, which is to overcome an MITB attack.

6.3 Future Work

We anticipate that the proposed authentication system can be extended to include an e-commerce payment. This may be achieved within the response generation phase and can include a secure platform that can execute an electronic data interchange for e-commerce transactions. Consequently, this extension will provide an integral authentication and transaction framework that overcomes the MITB attacks by using a QR code with mobile phone. The entities involved will include the server, mobile phone and the financial organization (bank) that completes the transaction. The steps involved in extending the existing system can be summarized as follows:

- In the challenge generation phase, the QR code contents will be expanded to include (1) the transaction information, and (2) the existing authentication information that was used to authenticate the client in this thesis. The transaction information will include a Transaction identification number ($\boldsymbol{TID}$), an Amount to transfer ($\boldsymbol{AMT}$), the Destination account ($\boldsymbol{DST}$) or the merchant, and the Source account ($\boldsymbol{SRC}$) or the bank.

- Following user authentication, the mobile application will submit the QR code-recovered $\boldsymbol{SRC}$ within the response generation phase in order for the server to initiate a secure SSL/TLS connection with the bank. The server will submit the client's Mobile number ($\boldsymbol{mn}$) along with the transaction information, which includes the $\boldsymbol{TID}$, $\boldsymbol{AMT}$ and $\boldsymbol{DST}$.

- The bank uses the $\boldsymbol{mn}$ received as a reference to retrieve client information and ensure that the client is a customer. Before performing the transaction, the bank verifies the received transaction information with the client. This verification sends the $\boldsymbol{TID}$, $\boldsymbol{AMT}$ and $\boldsymbol{DST}$ to the client's mobile phone via the Short message service (SMS). The mobile phone then verifies the received transaction information comparing it to the information recovered from the QR code. Upon verification, the mobile phone submits a confirmation message to the bank that is performing the transaction.

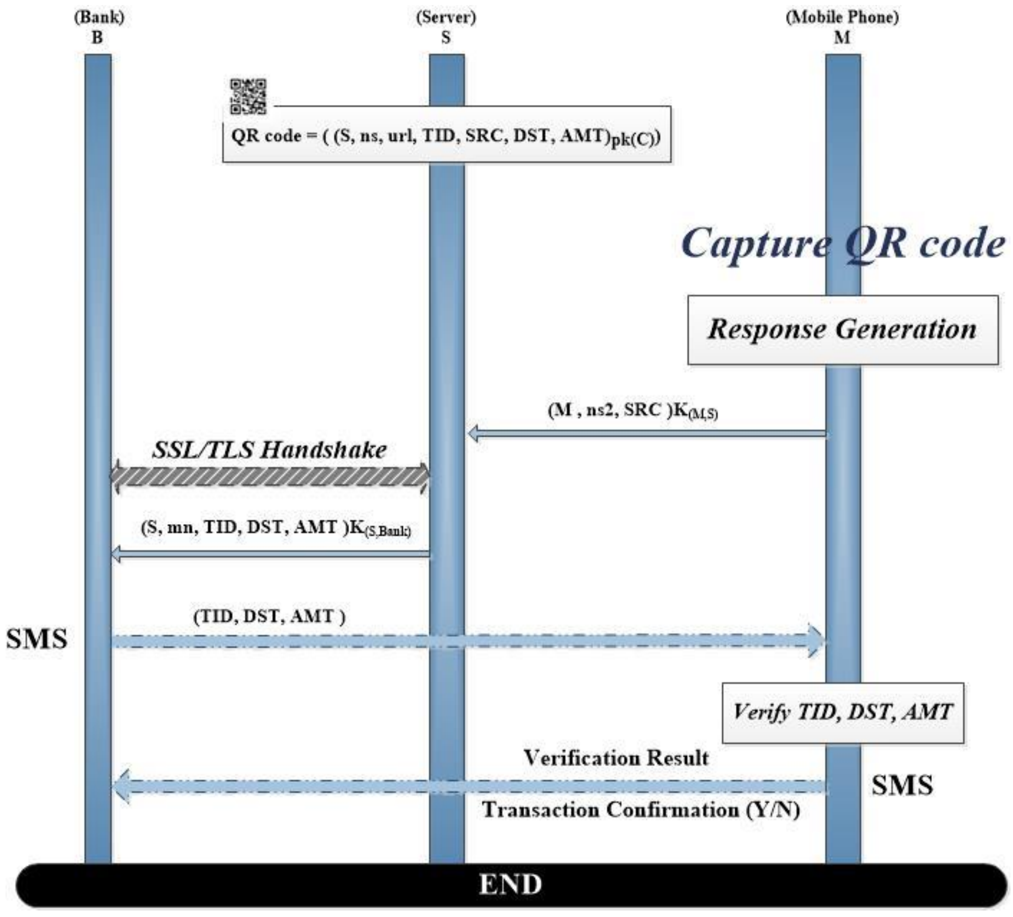


Figure 6.2: MUAS Future Transaction Mechanism

In terms of using the proposed MUAS approach in other disciplines within the context of client authentication, the ***challenge generation phase*** operates in the same manner, thus mitigating the Denial-of-Service attack (DOS) and generating the QR code challenge. The QR code is composed of the same contents, along with an additional account number $\boldsymbol{Account\_no}$ of the registered client. Upon capturing the QR code, the ***response generation phase*** operates in the same manner. Moreover, the mobile phone requests the client to submit his/her account number $\boldsymbol{Account\_Number}$ to be verified against the QR code-recovered account number $\boldsymbol{Account\_no}$. Upon verification, the mobile phone generates the response to be sent to the server for verification. In the case of a successful authentication, the initiating client is capable of accessing his/her personal records.

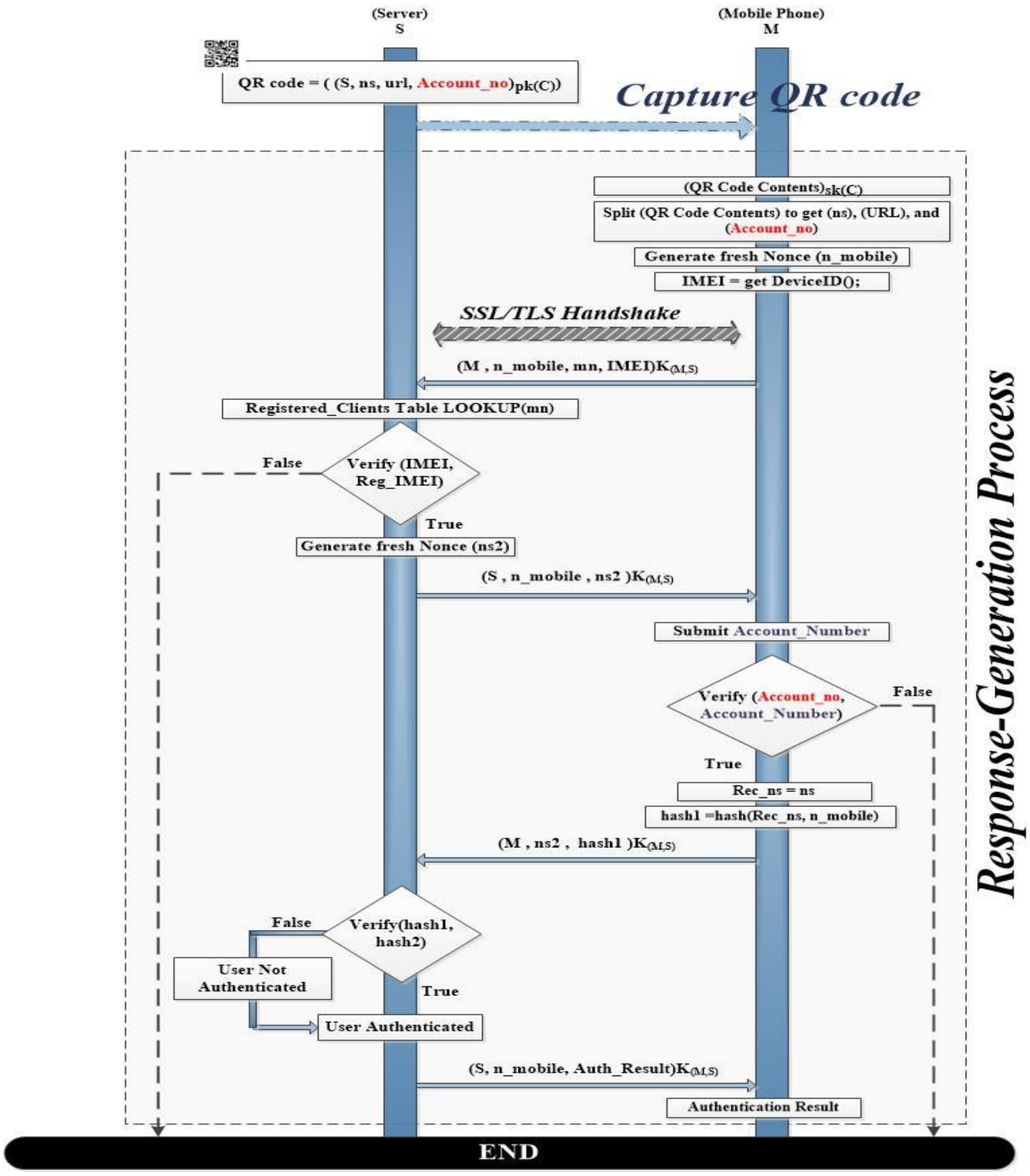

(Server) S
(Mobile Phone) M
QR code = ((S, ns, url, Account_no)pk(C))
Capture QR code
(QR Code Contents)sk(C)
Split (QR Code Contents) to get (ns), (URL), and (Account_no)
Generate fresh Nonce (n_mobile)
IMEI = get DeviceID();
SSL/TLS Handshake
(M , n_mobile, mn, IMEI)K(M,S)
Registered_Clients Table LOOKUP(mn)
False
Verify (IMEI, Reg_IMEI)
True
Generate fresh Nonce (ns2)
(S , n_mobile , ns2)K(M,S)
Submit Account_Number
Verify (Account_no, Account_Number)
False
True
Rec_ns = ns
hash1 =hash(Rec_ns, n_mobile)
(M , ns2 , hash1)K(M,S)
False
Verify(hash1, hash2)
User Not Authenticated
True
User Authenticated
(S, n_mobile, Auth_Result)K(M,S)
Authentication Result
END
Response-Generation Process

Figure 6.3: MUAS in Different Disciplines

6.4 MUAS Limitations

Although the proposed MUAS can overcome the problem of MITB attack within a secure framework that relocates the authentication mechanism from the infected browser to the secure mobile phone of the client, there are some limitations that are related to our proposed system, which we will briefly discuss.

- Confirming the legitimacy of the mobile phone holder within the challenge-generation phase requires a cellular network for submitting the OTP via SMS. Therefore, this step can be considered as a defect within the system in the case of a real-world threat, namely, an unavailable cellular network, thus causing the whole system to malfunction.
- In the challenge-generation phase, the basis for generating the QR code by the server relies upon verifying the received OTP from the presumably infected browser. Although the MITB attack's main concern is financial information, however if the OTP value was manipulated after user submission, the system will deny the authentication service which can cause denial of service attack. Another limitation within the response-generation phase states that the client is requested to use a smartphone only, for capturing and processing the QR code via a downloaded application.
- Another considered weakness of the proposed MUAS lies within the computational cost in terms of employing an expensive public key encryption algorithm to secure the QR code contents; this can be justified by the need to secure the code from unauthorized users. Moreover, the communication cost exceeds most related work in terms of number and size of exchanged messages. However, a compromise was made with respect to security.

Overall and despite the encountered limitations mentioned above, the proposed MUAS is a promising solution to provide a simple, secure, and low-energy consumption framework, which can prevent the Man-in-the-Browser (MITB) attack within a secure online authentication system for e-commerce applications. With this new algorithm, MUAS also excludes the risk of Denial-of-Service (DOS). In terms of security, the proposed solution satisfies the security properties required by all security protocols [41, 247]. Thus, this thesis have achieved its objectives and provided a novel and secure solution to one of the most sophisticated attacks, which is MITB attack.

References

[1] S. Rauti and V. Leppänen, "Chapter 28 - Man-in-the-Browser Attacks in Modern Web Browsers," in *Emerging Trends in ICT Security*, ed Boston: Morgan Kaufmann, 2014, pp. 469-480.

[2] M. Mannan, "Authentication and securing personal information in an untrusted internet," PhD, CARLETON UNIVERSITY Ottawa, 2009.

[3] P. Goyal, N. Bansal, and N. Gupta, "Averting man in the browser attack using user-specific personal images," in *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, 2013, pp. 1283-1286.

[4] F. B. M. Nor and K. A. Jalil, "An enhanced remote authentication scheme to mitigate man-in-the-browser attacks," in *Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on*, 2012, pp. 271-276.

[5] L. O'Gorman, "Comparing passwords, tokens, and biometrics for user authentication," *Proceedings of the IEEE*, vol. 91, pp. 2021-2040, 2003.

[6] Y. Jiangshan, W. Guilin, M. Yi, and G. Wei, "An Efficient Generic Framework for Three-Factor Authentication With Provably Secure Instantiation," *Information Forensics and Security, IEEE Transactions on*, vol. 9, pp. 2302-2313, 2014.

[7] H. Xinyi, X. Yang, A. Chonka, Z. Jianying, and R. H. Deng, "A Generic Framework for Three-Factor Authentication: Preserving Security and Privacy in Distributed Systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, pp. 1390-1397, 2011.

[8] C. K. Dimitriadis, "Analyzing the Security of Internet Banking Authentication Mechanisms " *Information System Control Journal* vol. 3, 2007.

[9] M. H. Eldefrawy, K. Alghathbar, and M. K. Khan, "OTP-Based Two-Factor Authentication Using Mobile Phones," in *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*, 2011, pp. 327-331.

[10] Emiliano De Cristofaro, Honglu Du, Julien Freudiger, and G. Norcie, "A Comparative Usability Study of Two-Factor Authentication," in *Proceedings of the Workshop on Usable Security (USEC)* 2014.

[11] K. Curran and T. Dougan, "Man in the Browser Attacks," *Internation Journal of Ambient Computation and Intelligenece*, vol. 4, pp. 29-39, 2012.

[12] R. K. K and R. Kumar, "Securing User Input as a Defense Against MitB," presented at the Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing, Amritapuri, India, 2014.

[13] S. T. Zargar, J. Joshi, and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, pp. 2046-2069, 2013.

[14] N. Utakrit, "Review of browser extensions, a man-in-the-browser phishing techniques targeting bank customers," in *Proceedings of the 7th Australian Information Security Management Conference, Perth, Western Australia, 1st to 3rd*, Edith Cowan University, 2009.

[15] S. Rauti, V. Lepp, #228, and nen, "Browser extension-based man-in-the-browser attacks against Ajax applications with countermeasures," presented at the Proceedings of the 13th International Conference on Computer Systems and Technologies, Ruse, Bulgaria, 2012.

[16] M. Niranjanamurthy and D. D. Chahar, "The study of e-commerce security issues and solutions," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 2, 2013.

[17] T. Lu and X. Lei, "Study on Security Framework in E-Commerce," in *2007 International Conference on Wireless Communications, Networking and Mobile Computing*, 2007, pp. 3541-3544.

[18] V. D. Sawma, "E-commerce security: A new methodology for deriving effective countermeasures design models," University of Ottawa (Canada), 2003.

[19] N. M. Al-Slamy, "E-Commerce security," *International Journal of Computer Science and Network Security IJCSNS,* vol. 8, pp. 340-344, 2008.

[20] G. Qingping, F. Li, and Y. Li, "Probe into E-Commerce Security Technology," in *2009 International Forum on Computer Science-Technology and Applications*, 2009, pp. 425-428.

[21] M. Stawowski, "Network Security Architecture," *ISSA Journal,* pp. 34-38, 2009.

[22] Z. Tian, N. Xu, and W. Peng, "E-Commerce Security: A Technical Survey," in *2008 Second International Symposium on Intelligent Information Technology Application*, 2008, pp. 956-960.

[23] Microsoft. (2017). *TCP/IP Protocol Architecture*. Available: https://technet.microsoft.com/en-us/library/cc958821.aspx

[24] K. C. Laudon, C. G. Traver, and A. V. R. Elizondo, *E-commerce* vol. 29: Pearson/Addison Wesley, 2007.

[25] R. Kissel, "Glossary of key information security terms," *NIST Interagency Reports NIST IR,* vol. 7298, 2013.

[26] William E. Burr, Donna F. Dodson, Elaine M. Newton, Ray A. Perlner, W. Timothy Polk, Sarbari Gupta, *et al.*, "Electronic Authentication Guideline " in *National Institute of Standards and Technology (NIST),* NIST Special Publication 800-63-1 ed, 2011.

[27] CISCO, "Cisco ME 2600X Series Ethernet Access Switch Software Configuration Guide ", ed: CISCO, 2013-2014, pp. 211-266.

[28] A. Basu and S. Muylle, "Authentication in e-commerce," *Commun. ACM,* vol. 46, pp. 159-166, 2003.

[29] Priti Jadhao and L. Dole, "Survey on Authentication Password Techniques " *International Journal of Soft Computing and Engineering (IJSCE)* vol. 3, pp. 67-68, May 2013.

[30] Saraswati B. Sahu and A. Singh, "Survey on Various Techniques of User Authentication and Graphical Password " *International Journal of Computer Trends and Technology (IJCTT),* vol. 16, pp. 98-102, Oct 2014.

[31] B.Sumitra, C.R. Pethuru, and M.Misbahuddin, "A Survey of Cloud Authentication Attacks and Solution Approaches " *International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE),* vol. 2, Oct 2014.

[32] Mudassar Raza, Muhammad Iqbal, Muhammad Sharif, and W. Haider, "A Survey of Password Attacks and Comparative Analysis on Methods for Secure Authentication," *World Applied Sciences Journal* vol. 19, pp. 439-444, 2012.

[33] J. A. and S. N.P., "A SURVEY ON AUTHENTICATION ATTACKS AND COUNTERMEASURES IN A DISTRIBUTED ENVIRONMENT " *Indian Journal of Computer Science and Engineering (IJCSE),* vol. 5, pp. 71-77, May 2014.

[34] Y. Hsiu-Lien, C. Tien-Ho, H. Kuei-Jung, and S. Wei-Kuan, "Robust elliptic curve cryptography-based three factor user authentication providing privacy of biometric data," *Information Security, IET,* vol. 7, pp. 247-252, 2013.

[35] M. El-Abed, P. Lacharme, and C. Rosenberger, "Privacy and Security Assessment of Biometric Systems," in *Advances in Security and Privacy of Biometric Systems*, p. Cambridge scholar, Ed., ed, 2015.

[36] M. Gomez-Barrero, J. Galbally, P. Tome, and J. Fierrez, "On the Vulnerability of Iris-Based Systems to a Software Attack Based on a Genetic Algorithm," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. vol. 7441, L. Alvarez, M. Mejail, L. Gomez, and J. Jacobo, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 114-121.

[37] H. S. Puri, "Vulnerability Assessment of Multi Biometric Systems," The Personal Risk Management Group Pty Ltd. March 31, 2015.

[38] B. Schneier, "Attack Trees," *Dr. Dobb's Journal,* vol. 24, pp. 21-29, December 1999.

[39] M. Johnson, "A new approach to Internet banking," University of Cambridge, Computer Laboratory, September 2008.

[40] L. Lamport, "Password authentication with insecure communication," *Commun. ACM,* vol. 24, pp. 770-772, 1981.

[41] Cas Cremers and Sjouke Mauw, *Operational Semantics and Verification of Security Protocols*: Springer, 2012.

[42] "The Secure Sockets Layer (SSL) Protocol Version 3.0 ", ed: Internet Engineering Task Force (IETF), RFC: 6101, August 2011.

[43] A. Mahboob and N. Ikram, "Transport Layer Security (TLS)--A Network Security Protocol for E-commerce," *Pakistan Navy Engineering College (PNEC) Research Journal*, 2004.

[44] T. Petsas, G. Tsirantonakis, E. Athanasopoulos, and S. Ioannidis, "Two-factor authentication: is the world ready?: quantifying 2FA adoption," presented at the Proceedings of the Eighth European Workshop on System Security, Bordeaux, France, 2015.

[45] A. E. Lackey, T. Pandey, M. Moshiri, N. Lalwani, C. Lall, and P. Bhargava, "Productivity, part 2: cloud storage, remote meeting tools, screencasting, speech recognition software, password managers, and online data backup," *Journal of the American College of Radiology,* vol. 11, pp. 580-588, 2014.

[46] Z. Li, W. He, D. Akhawe, and D. Song, "The emperor? s new password manager: Security analysis of web-based password managers," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014.

[47] K. Krol, E. Philippou, E. De Cristofaro, and M. A. Sasse, ""They brought in the horrible key ring thing!" Analysing the Usability of Two-Factor Authentication in UK Online Banking," *arXiv preprint arXiv:1501.04434,* 2015.

[48] A. Dmitrienko, C. Liebchen, C. Rossow, and A.-R. Sadeghi, "On the (in) security of mobile two-factor authentication," in *International Conference on Financial Cryptography and Data Security*, 2014, pp. 365-383.

[49] C. C. Chang and T. C. Wu, "Remote password authentication with smart cards," *Computers and Digital Techniques, IEE Proceedings E,* vol. 138, pp. 165-168, 1991.

[50] M. Ziqing, D. Florencio, and C. Herley, "Painless migration from passwords to two factor authentication," in *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on*, 2011, pp. 1-6.

[51] J. Choubey and B. Choubey, "Secure User Authentication in Internet Banking: A Qualitative Survey," *International Journal of Innovation, Management and Technology,* vol. 4, p. 198, 2013.

[52] HSBC. (2015). *Banking securely with Personal Internet Banking*. Available: https://www.us.hsbc.com/1/2/home/personal-banking/pib/security-device-logon-options

[53] Barclays. (2017). *Quick, safe and convenient - Online Banking made easy* [Online]. Available: https://bank.barclays.co.uk/olb/authlogin/loginAppContainer.do#/identification

[54] Barclays. (2017). *Guide to PINsentry, Protect your accounts* [Online]. Available: http://www.barclays.co.uk/ways-to-bank/online-banking/pinsentry-guide/

[55] HSBC. (2015). *Your Guide to the HSBC Security Device* [Online]. Available: https://www.us.hsbc.com/1/2/home/site/security/hsbc-online-banking-security

[56] First Direct. (2017). *Internet Banking* [Online]. Available: http://www1.firstdirect.com/1/2/banking/ways-to-bank/online-banking

[57] First Direct. (2017). *Secure Key* [Online]. Available: http://www1.firstdirect.com/1/2/securekey

[58] Nationwide. (2017). *Internet Banking Security, Keeping you safe and secure, Secure Banking* [Online]. Available: http://www.nationwide.co.uk/support/security-centre/internet-banking-security/how-we-keep-you-safe

[59] Nationwide. (2017). *Internet Banking Security, Card reader and security questions, Additional security with card readers and security questions* [Online]. Available: http://www.nationwide.co.uk/support/security-centre/internet-banking-security/card-reader-and-security-questions#xtab:using-a-card-reader

[60] Halifax. (Accessed 2017). *How we're protecting you.* [Online]. Available: http://www.halifax.co.uk/aboutonline/security/protecting-you/

[61] Lloyds Bank. (Accessed 2017). *Using Internet Banking* [Online]. Available: https://www.lloydsbank.com/online-banking/internet-banking/using-internet-banking/logon-logout.asp

[62] NatWest. (Accessed 2017). *Our Security Measures* [Online]. Available: http://personal.natwest.com/personal/security-centre/how-we-protect-you.html

[63] NatWest. (Accessed 2017). *Online Banking* [Online]. Available: http://personal.natwest.com/personal/ways-to-bank/online-banking.html

[64] UIster Bank. (Accessed 2017). *Getting started on Anytime Banking.* [Online]. Available: http://digital.ulsterbank.ie/personal/ways-to-bank/getting-started-with-online-banking.html

[65] UIster Bank. (Accessed 2017, 2016). *Online Banking Card Reader.* [Online]. Available: http://digital.ulsterbank.ie/globals/security-centre/card-reader.html

[66] Royal Bank of Scotland. (Accessed 2017). *Digital Banking* [Online]. Available: http://personal.rbs.co.uk/personal/ways-to-bank/digital-banking.html

[67] Santander. (Accessed 2017, 2016). *Online Banking* [Online]. Available: http://www.santander.co.uk/uk/help-support/online-banking

[68] The co-operative Bank. (Accessed 2017, 2016). *Online Banking* [Online]. Available: http://www.co-operativebank.co.uk/onlinebanking

[69] Standard Chartered. (2016). *Online & Mobile Security* [Online]. Available: https://www.sc.com/global/security-tips/

[70] Bank of Scotland. (Accessed 2017). *How we protect you* [Online]. Available: http://www.bankofscotland.co.uk/securityandprivacy/security/how-we-protect-you.asp

[71] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," presented at the Proceedings of the 6th ACM conference on Computer and communications security, Kent Ridge Digital Labs, Singapore, 1999.

[72] Biometrics.gov. (Accessed February, 2016). *NSTC Subcommittee on Biometrics and Identity Management Room*. Available: http://www.biometrics.gov/nstc/

[73] Biometrics.gov. (Accesses February, 2016). *Introduction to Biometrics- Biometric Glossary*. Available: http://www.biometrics.gov/ReferenceRoom/Introduction.aspx

[74] M. Weizhi, D. S. Wong, S. Furnell, and Z. Jianying, "Surveying the Development of Biometric User Authentication on Mobile Phones," *Communications Surveys & Tutorials, IEEE,* vol. 17, pp. 1268-1293, 2015.

[75] K. Renaud, A. Hoskins, and R. von Solms, "Biometric identification: Are we ethically ready?," in *Information Security for South Africa (ISSA), 2015*, 2015, pp. 1-8.

[76] S. Prakash and P. Gupta, "Introduction," in *Ear Biometrics in 2D and 3D: Localization and Recognition*, ed Singapore: Springer Singapore, 2015, pp. 1-20.

[77] BIO-key. (2016). *BIO-key, Revolutionizing Authentication*. Available: http://www.bio-key.com/

[78] BIO-key. (2015). *SideSwipe Mini Fingerprint Reader*. Available: http://bio-key.com/products/fingerprint-readers/sideswipe

[79] BIO-key. (March 2015). *EcoID Compact Fingerprint Reader*. Available: http://bio-key.com/products/software-solutions/ecoid

[80] R. Clarke, "Human identification in information systems: Management challenges and public policy issues," *Information Technology & People,* vol. 7, pp. 6-37, 1994.

[81] K. Harmon. (2009) Can You Lose Your Fingerprints? *Scientific American*. Available: http://www.scientificamerican.com/article/lose-your-fingerprints/

[82] V. Mhaske and A. Patankar, "Multimodal biometrics by integrating fingerprint and palmprint for security," in *Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on*, 2013, pp. 1-5.

[83] R. L. Telgad, P. D. Deshmukh, and A. M. N. Siddiqui, "Combination approach to score level fusion for Multimodal Biometric system by using face and fingerprint," in *Recent Advances and Innovations in Engineering (ICRAIE), 2014,* 2014, pp. 1-8.

[84] R. Waugh. (December 2014) Biometrics – can your fingerprint be ‘copied’ from a normal photo? *We Live Security*. Available: http://www.welivesecurity.com/

[85] P. Olsen. (July 2012) Researchers Trick Iris Scanner With Forged Eye Images. *Forbes*. Available: http://www.forbes.com/sites/parmyolson/2012/07/26/researchers-trick-iris-scanner-with-forged-eye-images/#65df22464aab

[86] F. Cheng, "Security Attack Safe Mobile and Cloud-based One-time Password Tokens Using Rubbing Encryption Algorithm," *Mobile Networks and Applications,* vol. 16, pp. 304-336, 2011/06/01 2011.

[87] F. Aloul, S. Zahidi, and W. El-Hajj, "Two factor authentication using mobile phones," in *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*, 2009, pp. 641-644.

[88] C. Yoo, B.-T. Kang, and H. Kim, "Case study of the vulnerability of OTP implemented in internet banking systems of South Korea," *Multimedia Tools and Applications,* vol. 74, pp. 3289-3303, 2015/05/01 2015.

[89] S. Hamdare, V. Nagpurkar, and J. Mittal, "Securing SMS Based One Time Password Technique from Man in the Middle Attack," *arXiv preprint arXiv:1405.4828*, 2014.

[90] Y.-W. Chow, W. Susilo, M. Au, and A. Barmawi, "A Visual One-Time Password Authentication Scheme Using Mobile Devices," in *Information and Communications Security*. vol. 8958, L. C. K. Hui, S. H. Qing, E. Shi, and S. M. Yiu, Eds., ed: Springer International Publishing, 2015, pp. 243-257.

[91] K. Subashini and G. Sumithra, "Secure multimodal mobile authentication using one time password," in *Current Trends in Engineering and Technology (ICCTET), 2014 2nd International Conference on*, 2014, pp. 151-155.

[92] A. Tandon, R. Sharma, S. Sodhiya, and P. Vincent, "QR Code based secure OTP distribution scheme for Authentication in Net-Banking," *International Journal of Engineering & Technology,* pp. 0975-4024, 2013.

[93] K. Ramesh and S. Ramesh, "Implementing One Time Password based security mechanism for securing personal health records in cloud," in *Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on*, 2014, pp. 968-972.

[94] N. Thiranant and H. Lee, "A Design of e-Healthcare Authentication Framework with QR Code," *International Journal of Security & Its Applications,* vol. 8, pp. 79-86, 2014.

[95] E. Sediyono, K. I. Santoso, and Suhartono, "Secure login by using One-time Password authentication based on MD5 Hash encrypted SMS," in *Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on*, 2013, pp. 1604-1608.

[96] Barclays. (2015). *Guide to Pinsentry* [Online]. Available: http://www.barclays.co.uk/Helpsupport/UpgradetoPINsentry/P1242559314766

[97] NatWest. (2015). *Security, See the many ways we protect you*. Available: http://personal.natwest.com/global/security-centre.html

[98] D. Moholkar, N. Kadam, D. Deokar, A. Kute, and S. Kadam, "An Efficient Approach for Phishing Website Detection using Visual Cryptography (VC) and Quick Response Code (QR code)," *International Journal of Computer Applications*, vol. 115, 2015.

[99] J. Shamdasani and P. Matte, "ATM Client Authentication System Using Biometric Identifier & OTP," *International Journal of Engineering Trends and Technology (IJETT),* vol. 11, pp. 255-258, 2014.

[100] C. Mulliner, R. Borgaonkar, P. Stewin, and J.-P. Seifert, "SMS-based one-time passwords: attacks and defense," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, ed: Springer, 2013, pp. 150-159.

[101] M. Toorani and A. Beheshti, "Solutions to the GSM Security Weaknesses," in *Next Generation Mobile Applications, Services and Technologies, 2008. NGMAST '08. The Second International Conference on*, 2008, pp. 576-581.

[102] N. Saxena and N. S. Chaudhari, "EasySMS: A Protocol for End-to-End Secure Transmission of SMS," *Information Forensics and Security, IEEE Transactions on,* vol. 9, pp. 1157-1168, 2014.

[103] S. Liu and S. Zhu, "A Novel QR Code and mobile phone based Authentication protocol via Bluetooth," presented at the International Conference on Materials Engineering and Information Technology Applications (MEITA), 2015.

[104] O. Leung. (2016). *Chapter 1: What is Mobile Instant Messaging?* . Available: https://dzone.com/articles/chapter-1-what-is-mobile-instant-messaging

[105] E. Bønes, P. Hasvold, E. Henriksen, and T. Strandenæs, "Risk analysis of information security in a mobile instant messaging and presence system for healthcare," *International Journal of Medical Informatics,* vol. 76, pp. 677-687, 2007/09/01/ 2007.

[106] R. B. Jennings, E. M. Nahum, D. P. Olshefski, D. Saha, S. Zon-Yin, and C. Waters, "A study of Internet instant messaging and chat protocols," *IEEE Network,* vol. 20, pp. 16-21, 2006.

[107] D. Lane, "Instant Messaging Security," MSc, Information Technology Security, University of Westminster, December 2003.

[108] G. Fenu and P. L. Pau, "An Analysis of Features and Tendencies in Mobile Banking Apps," *Procedia Computer Science,* vol. 56, pp. 26-33, 2015.

[109] K.-H. Ha, A. Canedoli, A. W. Baur, and M. Bick, "Mobile banking—insights on its increasing relevance and most common drivers of adoption," *Electronic Markets,* vol. 22, pp. 217-227, 2012.

[110] B. Panja, D. Fattaleh, M. Mercado, A. Robinson, and P. Meharia, "Cybersecurity in banking and financial sector: Security analysis of a mobile banking application," in *Collaboration Technologies and Systems (CTS), 2013 International Conference on*, 2013, pp. 397-403.

[111] W. He, X. Tian, J. Shen, and Y. Li, "Understanding Mobile Banking Applications' Security risks through Blog Mining and the Workflow Technology," 2015.

[112] H. Lee, Y. Zhang, and K. L. Chen, "An Investigation of Features and Security in Mobile Banking Strategy," *Journal of International Technology and Information Management*, vol. 22, p. 2, 2013.

[113] HSBC. (2015). *Your Guide to the HSBC Digital Security Device* [Online]. Available: https://www.us.hsbc.com/1/2/home/site/security/hsbc-online-banking-security

[114] HSBC. (2016). *Mobile Banking* [Online]. Available: https://www.hsbc.com.tr/eng/retail_banking/direct_banking/mobilebanking.asp

[115] First Direct. (2012). *Banking on the go App* [Online]. Available: http://www1.firstdirect.com/1/2/banking/ways-to-bank/mobile-banking-app

[116] Nationwide. (2015). *Mobile Banking*. Available: http://www.nationwide.co.uk/support/ways-to-bank/mobile-banking/help-what-you-can-do-and-faqs#tab:WhatyoucandoandampFAQs

[117] Halifax. (Accessed 2016). *Download Our Apps.* [Online]. Available: http://www.halifax.co.uk/aboutonline/download-apps/#Benefits

[118] LLoyds Bank. (Accessed 2016). *Mobile Banking App* [Online]. Available: https://www.lloydsbank.com/online-banking/mobile-banking/mobile-app.asp

[119] NatWest. (Accessed 2016). *Banking with our Mobile App* [Online]. Available: http://personal.natwest.com/personal/ways-to-bank-with-us/Mobile-app-new.html

[120] Uister Bank. (Accessed 2016, 2016). *Mobile Banking* [Online]. Available: http://digital.ulsterbank.ie/personal/ways-to-bank/mobile-banking/key-benefits-of-the-app.html

[121] Royal Bank of Scotland. (Accessed 2016, 2016). *Banking with our Mobile app* [Online]. Available: http://personal.rbs.co.uk/personal/ways-to-bank-with-us/mobile-banking/new-to-mobile-app.html

[122] Santander. (Accessed 2016, 2016). *Mobile Banking* [Online]. Available: http://www.santander.co.uk/uk/help-support/mobile-banking

[123] The co-operative Bank. (Accessed 2016, 2016). *Mobile Banking from The Co-operative Bank* [Online]. Available: http://www.co-operativebank.co.uk/mobile?int_cmp=tn1_ob_col2_row1-2_download-app_10032015

[124] Standard Chartered. (2014). *Standard Chartered Apps* [Online]. Available: https://www.sc.com/en/online-banking/apps/

[125] Bank of Scotland. (Accessed 2016). *Internet and Mobile Banking* [Online]. Available: https://www.bankofscotland.co.uk/privatebanking/services/internet-and-mobile-banking/

[126] A. Sen and Y. Jou, "Secure Data Transmission Technique for iPhone using Quick Response (QR) Code," *International Conference On"Multidisciplinary Innovation In Business Engineering Science & Technology" (MI-BEST)*, vol. 1, pp. 53-62, 2015.

[127] P. Kieseberg, M. Leithner, M. Mulazzani, L. Munroe, S. Schrittwieser, M. Sinha, *et al.*, "QR code security," presented at the Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia, Paris, France, 2010.

[128] C. Ugwu and T. Mesigo, "A Novel Mobile Wallet Based on Android OS and Quick Response Code Technology," *International Journal of Advanced Researchin Computer Science and Technology (IJARCST)*, vol. 3, pp. 85-89, 2015.

[129] J. Malik, D. Girdhar, R. Dahiya, and G. Sainarayanan, "Multifactor Authentication Using a QR Code and a One-Time Password," *Journal of Information Processing Systems*, vol. 10, 2014.

[130] A. Vapen, D. Byers, and N. Shahmehri, "2-clickauth optical challenge-response authentication," in *Availability, Reliability, and Security, 2010. ARES'10 International Conference on*, 2010, pp. 79-86.

[131] K. Choi, C. Lee, W. Jeon, K. Lee, and D. Won, "A mobile based anti-phishing authentication scheme using QR code," in *Mobile IT Convergence (ICMIC), 2011 International Conference on*, 2011, pp. 109-113.

[132] DENSO Wave Incorporated. (January, 2016). *QR Code Standardization*. Available: http://www.qrcode.com/en/about/standards.html

[133] B. Sago, "The Usage Level and Effectiveness of Quick Response (QR) Codes for Integrated Marketing Communication Purposes among College Students," *International Journal of Integrated Marketing Communications*, vol. 3, 2011.

[134] J.-H. Jung, R. Somerstein, and E. S. Kwon, "SHOULD I SCAN OR SHOULD I GO?: YOUNG CONSUMERS'MOTIVATIONS FOR SCANNING QR CODE ADVERTISING," *International Journal of Mobile Marketing*, vol. 7, 2012.

[135] K. Krombholz, P. Frühwirt, P. Kieseberg, I. Kapsalis, M. Huber, and E. Weippl, "QR code security: A survey of attacks and challenges for usable security," in *Human Aspects of Information Security, Privacy, and Trust*, ed: Springer, 2014, pp. 79-90.

[136] Y. S. Lee, N. H. Kim, H. Lim, H. Jo, and H. J. Lee, "Online banking authentication system using mobile-OTP with QR-code," in *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*, 2010, pp. 644-648.

[137] J. Lee, C.-H. Cho, and M.-S. Jun, "Secure quick response-payment (QR-Pay) system using mobile device," in *Advanced Communication Technology (ICACT), 2011 13th International Conference on*, 2011, pp. 1424-1427.

[138] G. Starnberger, L. Froihofer, and K. M. Göschka, "QR-TAN: Secure mobile transaction authentication," in *Availability, Reliability and Security, 2009. ARES'09. International Conference on*, 2009, pp. 578-583.

[139] DENSO Wave Incorporated. (January, 2016). *Information Capacity nad Versions of the QR Code*. Available: http://www.qrcode.com/en/about/version.html

[140] DENSO Wave Incorporated. (January, 2016). *Error Correction Feature*. Available: http://www.qrcode.com/en/about/error_correction.html

[141] DENSO Wave Incorporated. (January, 2016). *What is a QR Code?* Available: http://www.qrcode.com/en/about/

[142] I. Kapsalis, "security of QR codes," Master, Faculty of Information Technology, Mathematics and Electrical Engineering - Department of Telematics, Norwegian University of Science and Technology, Institutt for telematikk, 2013.

[143] D. Chyi-Ren, L. Yu-Hong, J. Liao, Y. Hao-Wei, and K. Wei-Luen, "A Location-based Mobile Advertisement Publishing System for Vendors," in *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*, 2011, pp. 24-29.

[144] REPORTERS WITHOUT BORDERS. (Accessed 12 January, 2016). *Reporters Without Borders, The Freedom of Information* [Online]. Available: http://en.rsf.org/

[145] T. Nudd. (April27, 2011) Print Campaigns That Really Speak to You QR codes get dictators talking in free-press appeal. *Adweek*. Available: http://www.adweek.com/adfreak/print-campaigns-really-speak-you-131046

[146] "QR Codes in the Media: Reporters without Borders," in *The QR Place*, ed, April30, 2011.

[147] QR Pay Limited. (2011). *QR Pay*. Available: http://www.qrpay.com/

[148] A. Adsul, A. Shukla, J. Sinojia, R. Sinkar, and S. Jagtap, "Secure Authentication Method using QR Code for Banking," *International Journal*, vol. 3, 2015.

[149] V. Kale, Y. Nakat, S. Bhosale, A. Bandal, and R. G. Patole, "A Mobile Based Authentication Scheme Using QR Code for Bank Security," *International Journal of Advanced Researchin Computer Science and Management Studies (IJARCSMS),* vol. 3, pp. 192-196, 2015.

[150] P. Kieseberg, S. Schrittwieser, M. Leithner, M. Mulazzani, E. Weippl, L. Munroe*, et al.*, "Malicious Pixels Using QR Codes as Attack Vector," in *Trustworthy Ubiquitous Computing*. vol. 6, I. Khalil and T. Mantoro, Eds., ed: Atlantis Press, 2012, pp. 21-38.

[151] ISO/IEC 18004, "Information technology — Automatic identification and data capture techniques — Bar code symbology — QR Code," ed, 15-6-2000.

[152] CISCO, "Next Generation Encryption " October 2015.

[153] Hans Delfs and Helmut Knebl, *Introduction to Cryptography Principles and Applications* Third ed.: Springer, 2015.

[154] M. A. Alia, A. A. Tamimi, and O. N. AL-Allaf, "Cryptography Based Authentication Methods," in *Proceedings of the World Congress on Engineering and Computer Science*, 2014.

[155] E. B. Barker, W. C. Barker, and A. Lee, *Guideline for implementing cryptography in the federal government*: US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2005.

[156] Valtteri Niemi and Kaisa Nyberg, "The Science of Cryptology," in *UMTS Security,* ed: Wiley, 2003, pp. 113-118.

[157] Christof Paar and Jan Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*: Springer, 2009.

[158] S. Pal, K. K. Soundra Pandian, and K. C. Ray, "FPGA implementation of stream cipher using Toeplitz Hash function," in *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, 2014, pp. 1834-1838.

[159] G. Singh and A. Supriya, "A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security," *International Journal of Computer Applications,* vol. 67, pp. 33-38, 2013.

[160] Czesław Kościelny, Mirosław Kurkowski , and Marian Srebrny, *Modern Cryptography Primer Theoretical Foundations and Practical Applications*: Springer, 2013.

[161] A. A. Zadeh and H. M. Heys, "Theoretical simple power analysis of the grain stream cipher," in *Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on*, 2013, pp. 1-5.

[162] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, "Block Ciphers," in *Handbook of Applied Cryptography,* ed: CRC Press, 1996.

[163] Lars R. Knudsen and Matthew J.B. Robshaw, *The Block Cipher Companion*: Springer, 2011.

[164] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, "Stream Ciphers," in *Handbook of Applied Cryptography,* ed: CRC Press, 1996.

[165] M. Ebrahim, S. Khan, and U. B. Khalid, "Symmetric algorithm survey: a comparative analysis," *International Journal of Computer Applications,* vol. 61, pp. 12-19, 2014.

[166] S. Kandola, "A Survey of Cryptographic Algorithms," St. Lawrence University, 2013.

[167] "Guidelines for the Selection and Use of Transport Layer Security (TLS) Implementations," ed: NIST Special Publication 800-52, June 2005.

[168] W. Diffie and M. E. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on,* vol. 22, pp. 644-654, 1976.

[169] M. Dubey and M. Y. Yadav, "Comparative Analysis of Cryptographic Algorithms," *International Journal of Engineering Technology Science and Research (IJETSR)*, vol. 2, 2015.

[170] E. G. Nithya and D. P. Raj, "Survey on Asymmetric Key Cryptography Algorithms," *Journal of advanced computing and communication technologies*, vol. 2, 2014.

[171] T. Dierks, "The transport layer security (TLS) protocol version 1.2," 2008.

[172] R. Gelashvili, "Attacks on re-keying and renegotiation in Key Exchange Protocols," Bachelor Thesis, ETH Zurich, 2012.

[173] A. H. Al Hadi, "A Survey On Some Encryption Algorithms And Verification Of RSA Technique," *International Journal of Scientific and Technology Research* vol. 2, pp. 285-287, December 2013.

[174] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*: CRC press, 1996.

[175] Q. Dang, *Recommendation for applications using approved hash algorithms*: US Department of Commerce, National Institute of Standards and Technology, 2012.

[176] K. C. Laudon, C. G. Traver, and A. V. R. Elizondo, *E-commerce: : business, technology, society*, Fourth ed.: London : Person Prentice Hall 2007.

[177] R. Purohit, U. Mishra, and A. Bansal, "A Survey on Recent Cryptographic Hash Function Designs," *International Journal of Emerging Trends and Technology in Computer Science (IJETTCS)*, vol. 2, pp. 117-122, February 2013.

[178] X. Wang, Y. L. Yin, and H. Yu, "Finding Collisions in the Full SHA-1," in *Advances in Cryptology – CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005. Proceedings*, V. Shoup, Ed., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 17-36.

[179] Y. Wang, G. Wang, and H. Zhang, "Random Number Generator Based on Hopfield Neural Network and SHA-2 (512)," in *Advancing Computing, Communication, Control and Management*, Q. Luo, Ed., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 198-205.

[180] M. Stevens, "New Collision Attacks on SHA-1 Based on Optimal Joint Local-Collision Analysis," in *Advances in Cryptology – EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, T. Johansson and P. Q. Nguyen, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 245-261.

[181] National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)," in *Draft Federal Information Processing Standards Publication 180-4* ed. Gaithersburg, 2011.

[182] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *Fast Software Encryption*, 2004, pp. 371-388.

[183] I. Karabey and G. Akman, "A cryptographic approach for secure client - server chat application using public key infrastructure (PKI)," in *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2016, pp. 442-446.

[184] S. Kim, H.-T. Oh, and Y.-G. Kim, "Certificate sharing system for secure certificate distribution in mobile environment," *Expert Systems with Applications*, vol. 44, pp. 67-77, 2016.

[185] S. Pichumani and S. K. Kasera, "On implementing security at the transport layer," in *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, 2008, pp. 318-326.

[186] E. Reimers, "On the security of TLS and IPsec: Mitigation through physical constraints," Institutionen för datavetenskap-Department of Computer and Information Science, Sweden2015.

[187] I. Kotuliak, P. Rybar, and P. Truchly, "Performance comparison of IPsec and TLS based VPN technologies," in *Emerging eLearning Technologies and Applications (ICETA), 2011 9th International Conference on*, 2011, pp. 217-221.

[188] V. R. Hiran, "Usability evaluation of IPsec configuring components," Master Thesis, Department of Computer and Information Science, Linköpings universitet Sweden, October 2015.

[189] T. Polk, K. McKay, and S. Chokhani, "Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations," in *NIST Special Publication 800-52* vol. 800, ed, 2014, p. 59.

[190] "The Transport Layer Security (TLS) Protocol (Version 1.2)," ed: Network Working Group, RFC 5246, August 2008.

[191] Microsoft. (March 28, 2003). *How TLS/SSL Works*. Available: https://technet.microsoft.com/en-us/library/cc783349(v=ws.10).aspx

[192] "Transport Layer Security (TLS)," in *technopedia*, ed, 2016.

[193] National Institute of Standards and Technology (NIST), "Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations," in *Release of NIST Special Publication 800-52 Revision 1*, Kerry McKay, Kim Quill, and G. Witte, Eds., ed, 2014.

[194] P. Eronen and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)," *Network Working Group, RFC 4346*, 2005.

[195] S. Vandeven, "Ssl/tls: What’s under the hood," *SANS Institute InfoSec Reading Room,* vol. 13, 2013.

[196] Chilkat Software Inc. (2014). *(C++) Diffie-Hellman Key Exchange (DH)*. Available: http://www.example-code.com/cpp/dh_key_exchange.asp

[197] L. S. Huang, S. Adhikarla, D. Boneh, and C. Jackson, "An Experimental Study of TLS Forward Secrecy Deployments," *IEEE Internet Computing,* vol. 18, pp. 43-51, 2014.

[198] N. Aaraj, A. Raghunathan, and N. K. Jha, "Analysis and design of a hardware/software trusted platform module for embedded systems," *ACM Trans. Embed. Comput. Syst.,* vol. 8, pp. 1-31, 2009.

[199] N. Aaraj, A. Raghunathan, S. Ravi, and N. K. Jha, "Energy and Execution Time Analysis of a Software-based Trusted Platform Module," in *2007 Design, Automation & Test in Europe Conference & Exhibition*, 2007, pp. 1-6.

[200] Y.-W. Chow, W. Susilo, G. Yang, M. H. Au, and C. Wang, "Authentication and Transaction Verification Using QR Codes with a Mobile Device," in *Security, Privacy, and Anonymity in Computation, Communication, and Storage: 9th International Conference, SpaCCS 2016, Zhangjiajie, China, November 16-18, 2016, Proceedings*, G. Wang, I. Ray, J. M. Alcaraz Calero, and S. M. Thampi, Eds., ed Cham: Springer International Publishing, 2016, pp. 437-451.

[201] B. Dodson, D. Sengupta, D. Boneh, and M. S. Lam, "Snap2Pass: Consumer-Friendly Challenge-Response Authentication with a Phone," *http:/prpl. stanford. edu/papers/soups10j. pdf,* 2010.

[202] R. Oppliger, R. Hauser, and D. Basin, "SSL/TLS session-aware user authentication–Or how to effectively thwart the man-in-the-middle," *Computer Communications,* vol. 29, pp. 2238-2246, 2006.

[203] IBM. (August 15, 2015). *Setting the user ID pattern for certificate authentication*. Available: https://www.ibm.com/support/knowledgecenter/SSCRJU_3.2.0/com.ibm.swg.im.infosphere.streams.admin.doc/doc/ibminfospherestreams-adminconsole-certificate-authentication.html

[204] IBM. (August 15, 2015). *Creating a distinguished name for a user account*. Available: https://www.ibm.com/support/knowledgecenter/SSTFWV_5.1.0/com.ibm.itim.doc/usr_winad_5135.htm

[205] Oracle. (2000). *Netscape Certificate Management System Administrator's Guide: Distinguished Names*. Available: https://docs.oracle.com/cd/E19957-01/816-5531-10/app_dn.htm#1019922

[206] IBM. (2012). *Types of certificate name filters*. Available: https://www.ibm.com/support/knowledgecenter/SSLTBW_1.13.0/com.ibm.zos.r13.icha700/cnftypes.htm

[207] Microsoft. (2017). *About two-step verification*. Available: https://support.microsoft.com/en-us/help/12408/microsoft-account-about-two-step-verification

[208] Google. *Google 2-Step Verification*. Available: https://www.google.com/landing/2step/#tab=why-you-need-it

[209] W. E. Capers, "Sorkel, Aqua, and Sailfish - A crash course in writing embedded servers," in *Embedded Web Server Development*, Version 2.3.0.4 ed, 2012.

[210] (1993-2016). *C++ Builder*. Available: https://www.embarcadero.com/products/cbuilder

[211] (1998). *Chilkat Software*. Available: http://www.chilkatsoft.com/

[212] *TBarCode SDK - ActiveX/DLL/.NET Barcode Generator*. Available: http://www.tec-it.com/en/software/barcode-software/tbarcode/barcode-generator/Default.aspx

[213] (1996). *Barcode Software, Label Software, Reporting Software, Data Capture, 2D Barcode Generator*. Available: http://www.tec-it.com/en/start/Default.aspx

[214] Chilkat Software Inc. (2000-2016). *CkSocket C++ Reference Documentation*. Available: https://www.chilkatsoft.com/refdoc/vcCkSocketRef.html

[215] Chilkat Software Inc. (2000-2016). *CkCert C++ Reference Documentation*. Available: https://www.chilkatsoft.com/refdoc/vcCkCertRef.html

[216] Chilkat Software Inc. (2000-2016). *(C++) SSL Server Example*. Available: https://www.example-code.com/cpp/ssl_server.asp

[217] Chilkat Software Inc. (2000-2016). *(C++) SSL Client Certificate*. Available: https://www.example-code.com/cpp/ssl_client_certificate.asp

[218] N. Ferguson, B. Schneier, and T. Kohno, *Cryptography engineering: design principles and practical applications*: John Wiley & Sons, 2011.

[219] Chilkat Software Inc. (2000-2016). *CkPrng C++ Reference Documentation*. Available: https://www.chilkatsoft.com/refdoc/vcCkPrngRef.html

[220] Chilkat Software Inc. (2000-2016). *(C++) Fortuna PRNG Generate Random Encoded*. Available: https://www.example-code.com/cpp/prng_generate_random.asp

[221] Chilkat Software Inc. (2000-2016). *CkCertStore C++ Reference Documentation*. Available: https://www.chilkatsoft.com/refdoc/vcCkCertStoreRef.html

[222] Chilkat Software Inc. (2000-2016). *CkString C++ Reference Documentation*. Available: https://www.chilkatsoft.com/refdoc/vcCkStringRef.html

[223] Chilkat Software Inc. (2000-2016). *(C++) Export Digital Certificate's Public Key*. Available: https://www.example-code.com/cpp/cert_export_public_key.asp

[224] Chilkat Software Inc. (2000-2016). *CkPublicKey C++ Reference Documentation*. Available: https://www.chilkatsoft.com/refdoc/vcCkPublicKeyRef.html

[225] Chilkat Software Inc. (2000-2016). *CkRsa C++ Reference Documentation*. Available: https://www.chilkatsoft.com/refdoc/vcCkRsaRef.html

[226] Chilkat Software Inc. (2000-2016). *(C++) RSA Encrypt and Decrypt Strings*. Available: https://www.example-code.com/cpp/rsa_encryptStrings.asp

[227] TEC_IT, "TBarCode Library - Barcode Generation Library - Developer Manual," Version 11 ed, 2014.

[228] TEC-IT, "TEC-IT Barcode Software - Barcode Overview - Reference," Version 11 ed, 2015.

[229] Apple Developer. (2015). *Accessing a Bundle's Contents*. Available: https://developer.apple.com/library/content/documentation/CoreFoundation/Conceptual/CFBundles/AccessingaBundlesContents/AccessingaBundlesContents.html

[230] Apple Developer. (2016). *NSData - Foundation | Apple Developer Documentation*. Available: https://developer.apple.com/reference/foundation/nsdata#overview

[231] Chilkat Software Inc. (2000-2016). *CkoCert Objective-C Reference Documentation*. Available: https://www.chilkatsoft.com/refdoc/objcCkoCertRef.html

[232] Chilkat Software Inc. (2000-2016). *CkoPrivateKey Objective-C Reference Documentation*. Available: https://www.chilkatsoft.com/refdoc/objcCkoPrivateKeyRef.html

[233] Chilkat Software Inc. (2000-2016). *CkoRsa Objective-C Reference Documentation*. Available: https://www.chilkatsoft.com/refdoc/objcCkoRsaRef.html

[234] Chilkat Software Inc. (2000-2016). *(Objective-C) RSA Encrypt and Decrypt Strings*. Available: https://www.example-code.com/objc/rsa_encryptStrings.asp

[235] Apple Developer. (2016). *NSString Foundation - Apple Developer Documentation*. Available: https://developer.apple.com/reference/foundation/nsstring

[236] Chilkat Software Inc. (2000-2016). *(Objective-C) SSL Client Certificate*. Available: https://www.example-code.com/objc/ssl_client_certificate.asp

[237] Chilkat Software Inc. (2000-2016). *CkoCrypt2 Objective-C Reference Documentation*. Available: https://www.chilkatsoft.com/refdoc/objcCkoCrypt2Ref.html

[238] Chilkat Software Inc. (2000-2016). *(Objective-C) Hash Algorithms: SHA-1, HAVAL, MD2, MD5, SHA-256, SHA-384, SHA-512*. Available: https://www.example-code.com/objc/crypt_hash_algorithms.asp

[239] N. Dalal, J. Shah, K. Hisaria, and D. Jinwala, "A comparative analysis of tools for verification of security protocols," *Int'l J. of Communications, Network and System Sciences,* vol. 3, p. 779, 2010.

[240] C. Cremers, "The Scyther Tool," v1.1.3 ed, April 4, 2014.

[241] K. Pfeffer, "Formal Verification of a LTE Security Protocol for Dual-Connectivity: An Evaluation of Automatic Model Checking Tools," 2014.

[242] David Basin, Cas Cremers, and C. Meadows, *Model Checking Security Protocols*: Springer, October 22, 2012.

[243] E. Zenner, "Nonce Generators and the Nonce Reset Problem," in *Information Security*. vol. 5735, P. Samarati, M. Yung, F. Martinelli, and C. Ardagna, Eds., ed: Springer Berlin Heidelberg, 2009, pp. 411-426.

[244] B. Schneier, "New Attack on AES," in *Schneier on Security,* ed, Augest 18, 2011.

[245] R. Z. ITU-T and Z. Recommendation, "120: Message sequence chart (MSC)," *ITU-T, Geneva*, vol. 27, 1996.

[246] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha, "Analyzing the energy consumption of security protocols," presented at the Proceedings of the 2003 international symposium on Low power electronics and design, Seoul, Korea, 2003.

[247] C. J. F. Cremers, "Scyther - Semantics and Verication of Security Protocols," PhD Thesis, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, 2006.

[248] A. W. Roscoe, "Intensional specifications of security protocols," in *Proceedings 9th IEEE Computer Security Foundations Workshop*, 1996, pp. 28-38.

[249] G. Lowe, "A hierarchy of authentication specifications," in *Computer Security Foundations Workshop, 1997. Proceedings., 10th*, 1997, pp. 31-43.

[250] C. J. Cremers, S. Mauw, and E. P. D. Vink, "Defining authentication in a trace model," in *Fast*, 2003, pp. 131-145.

[251] C. J. F. Cremers, S. Mauw, and E. P. de Vink, "A Syntactic Criterion for Injectivity of Authentication Protocols," *Electronic Notes in Theoretical Computer Science,* vol. 135, pp. 23-38, 7/5/ 2005.

[252] C. J. F. Cremers, S. Mauw, and E. P. de Vink, "Injective synchronisation: An extension of the authentication hierarchy," *Theoretical Computer Science,* vol. 367, pp. 139-161, 11/24/ 2006.

[253] N. Kahya, N. Ghoualmi, and P. Lafourcade, "Secure key management protocol in WIMAX," *International Journal,* vol. 4, 2012.

[254] R. Patel, B. Borisaniya, A. Patel, D. Patel, M. Rajarajan, and A. Zisman, "Comparative analysis of formal model checking tools for security protocol verification," in *International Conference on Network Security and Applications*, 2010, pp. 152-163.

[255] C. Cremers, *Scyther: Unbounded Verification of Security Protocols*, 2007.

[256] C. J. F. Cremers, "Unbounded verification, falsification, and characterization of security protocols by pattern refinement," presented at the Proceedings of the 15th ACM conference on Computer and communications security, Alexandria, Virginia, USA, 2008.

[257] C. Cremers, "Scyther User Manual," *Department of Computer Science, University of Oxford: Oxford, UK,* 2014.

[258] C. J. Cremers, "The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols," presented at the Proceedings of the 20th international conference on Computer Aided Verification, Princeton, NJ, USA, 2008.

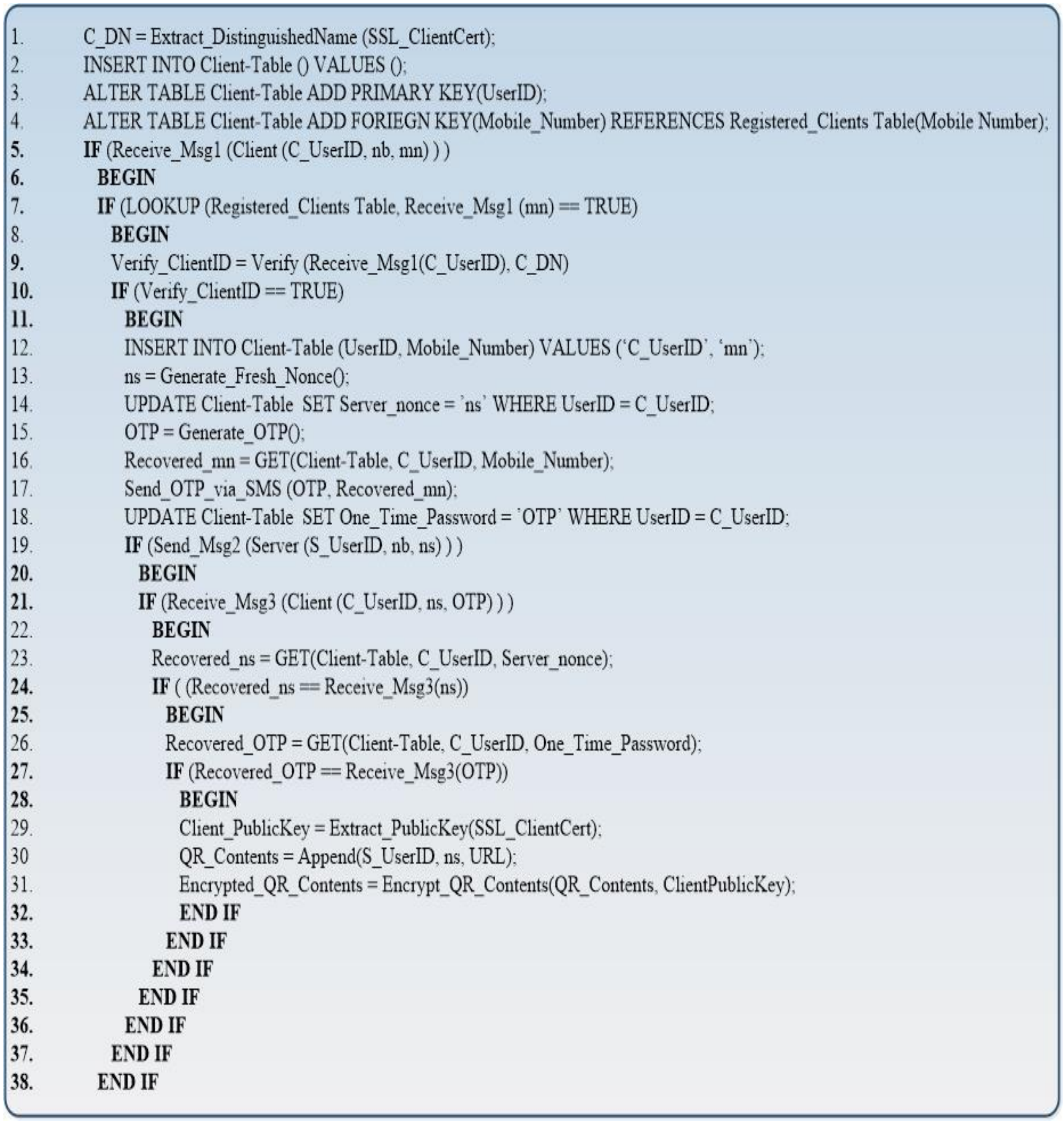
[259] S. Cleemput, G. Deconinck, Y. D. Mulder, K. Devos, B. Preneel, S. Seys*, et al.*, "FM-BIASED Applying the scyther formal verification tool on the DLMS/COSEM standard - Version 2.1 " 2014.

[260] P. Mitharwal, C. Lohr, and A. Gravey, "Survey on Network Interface Selection in Multihomed Mobile Networks," in *Advances in Communication Networking: 20th EUNICE/IFIP EG 6.2, 6.6 International Workshop, Rennes, France, September 1-5, 2014, Revised Selected Papers*, Y. Kermarrec, Ed., ed Cham: Springer International Publishing, 2014, pp. 134-146.

[261] C. Seongho, N. Jongkeun, and K. Chongkwon, "A dynamic load sharing mechanism in multihomed mobile networks," in *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, 2005, pp. 1459-1463 Vol. 3.

Appendix A

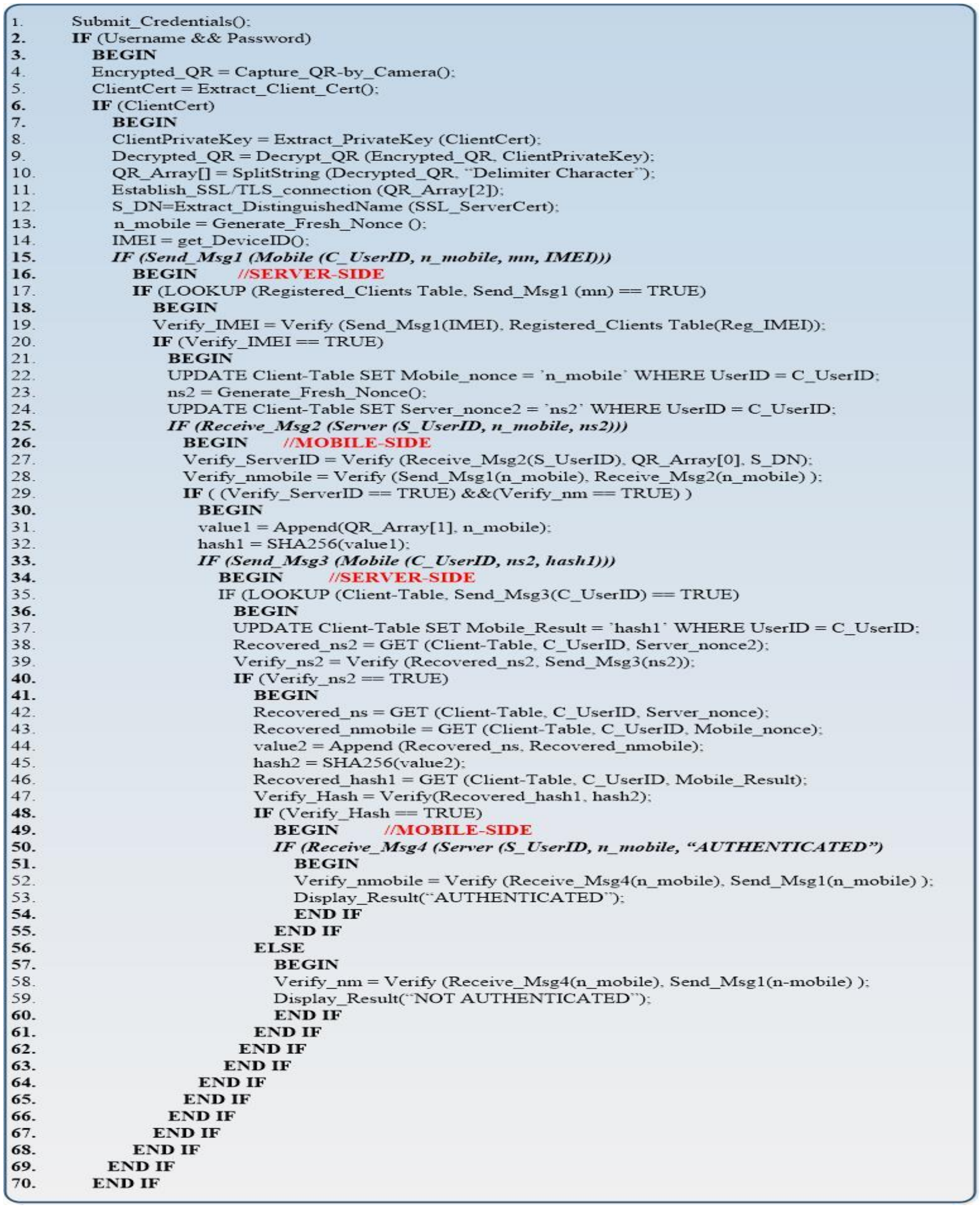
Pseudocode of the Challenge Generation Phase



```
C_DN = Extract_DistinguishedName (SSL_ClientCert);
INSERT INTO Client-Table () VALUES ();
ALTER TABLE Client-Table ADD PRIMARY KEY(UserID);
ALTER TABLE Client-Table ADD FORIEGN KEY(Mobile_Number) REFERENCES Registered_Clients Table(Mobile Number);
IF (Receive_Msg1 (Client (C_UserID, nb, mn) ) )
  BEGIN
  IF (LOOKUP (Registered_Clients Table, Receive_Msg1 (mn) == TRUE)
    BEGIN
    Verify_ClientID = Verify (Receive_Msg1(C_UserID), C_DN)
    IF (Verify_ClientID == TRUE)
      BEGIN
      INSERT INTO Client-Table (UserID, Mobile_Number) VALUES ('C_UserID', 'mn');
      ns = Generate_Fresh_Nonce();
      UPDATE Client-Table  SET Server_nonce = 'ns' WHERE UserID = C_UserID;
      OTP = Generate_OTP();
      Recovered_mn = GET(Client-Table, C_UserID, Mobile_Number);
      Send_OTP_via_SMS (OTP, Recovered_mn);
      UPDATE Client-Table  SET One_Time_Password = 'OTP' WHERE UserID = C_UserID;
      IF (Send_Msg2 (Server (S_UserID, nb, ns) ) )
        BEGIN
        IF (Receive_Msg3 (Client (C_UserID, ns, OTP) ) )
          BEGIN
          Recovered_ns = GET(Client-Table, C_UserID, Server_nonce);
          IF ( (Recovered_ns == Receive_Msg3(ns))
            BEGIN
            Recovered_OTP = GET(Client-Table, C_UserID, One_Time_Password);
            IF (Recovered_OTP == Receive_Msg3(OTP))
              BEGIN
              Client_PublicKey = Extract_PublicKey(SSL_ClientCert);
              QR_Contents = Append(S_UserID, ns, URL);
              Encrypted_QR_Contents = Encrypt_QR_Contents(QR_Contents, ClientPublicKey);
              END IF
            END IF
          END IF
        END IF
      END IF
    END IF
  END IF
```

Figure A.1: Challenge Generation Pseudo Code

Pseudocode of the Response Generation Phase



```
Submit_Credentials();
IF (Username && Password)
  BEGIN
  Encrypted_QR = Capture_QR-by_Camera();
  ClientCert = Extract_Client_Cert();
  IF (ClientCert)
    BEGIN
    ClientPrivateKey = Extract_PrivateKey (ClientCert);
    Decrypted_QR = Decrypt_QR (Encrypted_QR, ClientPrivateKey);
    QR_Array[] = SplitString (Decrypted_QR, "Delimiter Character");
    Establish_SSL/TLS_connection (QR_Array[2]);
    S_DN=Extract_DistinguishedName (SSL_ServerCert);
    n_mobile = Generate_Fresh_Nonce ();
    IMEI = get_DeviceID();
    IF (Send_Msg1 (Mobile (C_UserID, n_mobile, mn, IMEI)))
      BEGIN      //SERVER-SIDE
      IF (LOOKUP (Registered_Clients Table, Send_Msg1 (mn) == TRUE)
        BEGIN
        Verify_IMEI = Verify (Send_Msg1(IMEI), Registered_Clients Table(Reg_IMEI));
        IF (Verify_IMEI == TRUE)
          BEGIN
          UPDATE Client-Table SET Mobile_nonce = 'n_mobile' WHERE UserID = C_UserID;
          ns2 = Generate_Fresh_Nonce();
          UPDATE Client-Table SET Server_nonce2 = 'ns2' WHERE UserID = C_UserID;
          IF (Receive_Msg2 (Server (S_UserID, n_mobile, ns2)))
            BEGIN      //MOBILE-SIDE
            Verify_ServerID = Verify (Receive_Msg2(S_UserID), QR_Array[0], S_DN);
            Verify_nmobile = Verify (Send_Msg1(n_mobile), Receive_Msg2(n_mobile) );
            IF ( (Verify_ServerID == TRUE) &&(Verify_nm == TRUE) )
              BEGIN
              value1 = Append(QR_Array[1], n_mobile);
              hash1 = SHA256(value1);
              IF (Send_Msg3 (Mobile (C_UserID, ns2, hash1)))
                BEGIN      //SERVER-SIDE
                IF (LOOKUP (Client-Table, Send_Msg3(C_UserID) == TRUE)
                  BEGIN
                  UPDATE Client-Table SET Mobile_Result = 'hash1' WHERE UserID = C_UserID;
                  Recovered_ns2 = GET (Client-Table, C_UserID, Server_nonce2);
                  Verify_ns2 = Verify (Recovered_ns2, Send_Msg3(ns2));
                  IF (Verify_ns2 == TRUE)
                    BEGIN
                    Recovered_ns = GET (Client-Table, C_UserID, Server_nonce);
                    Recovered_nmobile = GET (Client-Table, C_UserID, Mobile_nonce);
                    value2 = Append (Recovered_ns, Recovered_nmobile);
                    hash2 = SHA256(value2);
                    Recovered_hash1 = GET (Client-Table, C_UserID, Mobile_Result);
                    Verify_Hash = Verify(Recovered_hash1, hash2);
                    IF (Verify_Hash == TRUE)
                      BEGIN      //MOBILE-SIDE
                      IF (Receive_Msg4 (Server (S_UserID, n_mobile, "AUTHENTICATED")
                        BEGIN
                        Verify_nmobile = Verify (Receive_Msg4(n_mobile), Send_Msg1(n_mobile) );
                        Display_Result("AUTHENTICATED");
                        END IF
                      END IF
                    ELSE
                      BEGIN
                      Verify_nm = Verify (Receive_Msg4(n_mobile), Send_Msg1(n-mobile) );
                      Display_Result("NOT AUTHENTICATED");
                      END IF
                    END IF
                  END IF
                END IF
              END IF
            END IF
          END IF
        END IF
      END IF
    END IF
  END IF
END IF
```

Figure 0.1: Response Generation Pseudo Code