January 2015

Design, Testing and Implementation of a New Authentication Method Using Multiple Devices

Cagri Cetin
University of South Florida, cagricetin@mail.usf.edu

Design, Testing and Implementation of a New Authentication Method Using Multiple Devices

by

Cagri Cetin

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Jay Ligatti, Ph.D.
Dmitry Goldgof, Ph.D.
Yao Liu, Ph.D.

Date of Approval:
May 26, 2015

Keywords: Authentication protocols, security, mobile devices, verification, access control

ACKNOWLEDGMENTS

I would like to express my special thanks to my advisor Dr. Jay Ligatti for being tremendous mentor for me. I would also like to thank Dr. Dmitry Goldgof for his support and advising during the project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Authentication protocols are very common mechanisms to confirm the legitimacy of someone's or something's identity in digital and physical systems.

This thesis presents a new and robust authentication method based on users' multiple devices. Due to the popularity of mobile devices, users are becoming more likely to have more than one device (e.g., smartwatch, smartphone, laptop, tablet, smart-car, smart-ring, etc.). The authentication system presented here takes advantage of these multiple devices to implement authentication mechanisms. In particular, the system requires the devices to collaborate with each other in order for the authentication to succeed. This new authentication protocol is robust against theft-based attacks on single device; an attacker would need to steal multiple devices in order to compromise the authentication system.

The new authentication protocol comprises an authenticator and at least two user devices, where the user devices are associated with each other. To perform an authentication on a user device, the user needs to respond a challenge by using his/her associated device. After describing how this authentication protocol works, this thesis will discuss three different versions of the protocol that have been implemented. In the first implementation, the authentication process is performed by using two smartphones. Also, as a challenge, a QR code is used. In the second implementation, instead of using a QR code, NFC technology is used for challenge transmission. In the last implementation, the usability with different platforms is exposed. Instead of using smartphones, a laptop computer and a smartphone combination is used. Furthermore, the authentication protocol has been verified by using an automated protocol-verification tool to check whether the protocol satisfies authenticity and

secrecy properties. Finally, these implementations are tested and analyzed to demonstrate the performance variations over different versions of the protocol.

CHAPTER 1

INTRODUCTION

Access control is a crucial component in many digital and physical systems in order to prevent unauthorized access to sensitive information. Access control policies are designed to ensure appropriate access to sensitive information and resources (e.g., smartphones, servers, smartwatches, operating systems, web applications). Hence, authentication methods are the key mechanism to enforce access control policies as well as protect access to secure systems.

1.1 Background

There are three standard approaches (factors) in authentication schemes:

- Authentication with something the user knows
- Authentication with something the user has
- Authentication with something the user is

A wide variety of authentication methods has been developed for access control systems. One of the most popular techniques is using a username and a password for authentication. Another common approach for extensive security is authentication with bio-metric identity (e.g.: fingerprint, retina). In addition to these techniques, multi-factor authentication is also commonly accepted by enterprise organizations.

1.1.1 Single-factor Authentication

Single-factor authentication is a cost-effective solution to implement access control mechanisms into the systems. Only one of the authentication factors needs to be used to implement single-factor authentication.

Using something the user knows, such as a username and a password, four-digit pin number, is a common solution in digital systems. For example, many of the web applications (e.g., email clients, social network applications) uses a username and a password authentication scheme [1]. Moreover, in mobile systems, four-digit pin number authentication is popular way to implement screen locks.

On the other hand, enforcing access control policies in physical systems are achieved by using something the user has. Physical tokens are a good example for these kind of authentication schemes [2]. For instance, ID cards could be used to unlock an office door or a garage gate. Furthermore, a credit or debit card are also examples of authentication by using something the user has.

Another way to implement an authentication method is fallowing the something the user is approach. Using the users' biometric identity is a common strategy for authentication protocols [3]. In some systems, the user needs to scan her/his fingerprint, retina or iris in order to prove her/his identity. In addition, face recognition techniques are considered in the domain of authentication schemes that use something the user is.

1.1.2 Multi-factor Authentication

Multi-factor authentication is another approach for implementing access control methods by using at least two of three authentication factors[1]. Introducing at least one more factor into the authentication process increases the difficulty of credential falsification.

In some systems, a fingerprint (something the user is) and a password (something the user knows) combination is used in order to implement multi-factor authentication. Similarly,

using a debit card (something the user has) and a pin number (something the user knows) is also common way to authenticate the users into banking systems.

However, the great concern with these existing authentication methods is the vulnerability against theft-based attacks [4]. In each scheme, the attacker could steal the identity information, such as a username and a password, a bio-metric identity or a token device, and access the users' sensitive information.

1.2 An Overview of the New Authentication Method

The new authentication method presented in this thesis is an innovative way to authenticate users by using users' multiple devices. This method uses at least two pre-associated (paired) devices to authenticate. The user devices need to cooperate with each other in order to respond to a challenge and successfully authenticate. Using associated devices is a robust way to prevent theft-based attacks on authentication protocols. Theoretical analysis has been done on the new authentication protocol in Jean-Baptiste Subils' thesis [5]. Additionally, a U.S. Utility Patent Application regarding the new authentication method was filed April 22, 2015 [6].

This thesis argues that the presented authentication protocol can be easily implemented to solve real-life authentication problems, and these implementations can have satisfactory performance. Furthermore, the authentication protocol can be model checked by a model checker to ensure that it satisfies secrecy and authenticity properties.

Throughout this thesis, terms "*first device*" and "*device one*" refer to the device that is attempting to authenticate some system. Terms "*second device*" and "*device two*" refer to the device with which the user is performing a task or a challenge to prove his/her identity. Also, the term "*challenge*" represents a required task to complete authentication.

After defining the new authentication method, this thesis will discuss three different versions of the protocol that have been designed and implemented. The main purpose of

introducing the different set of implementations is to demonstrate the new authentication protocol's adaptability to different authentication problems.

In the first implementation, the authentication process is performed by using two smartphones. Also, as a challenge, a QR code is used. When the first device attempts to access the system, the system sends a QR code to the device one. Then, the user needs to scan this QR code with his/her second smartphone in order to perform the challenge. Finally, the authenticator system decides whether or not the authentication is successful. Moreover, to demonstrate the authentication process with two smartphones and a QR code, an Android application was developed and installed into the smartphones. Also, a web server application was developed and deployed to represent the authenticator system.

In the second implementation, a different challenge transmission technique usage is demonstrated. Instead of using a QR code, NFC technology is used for challenge transmission. Similar to the first implementation, two smartphones, which run the Android application, and a web server application are used.

In the last implementation, the usability with different platforms is exposed. Instead of using smartphones, a laptop computer and a smartphone combination is used. In this particular implementation, the laptop attempts to access to sensitive information on some system. To perform an authentication, the authenticator system sends a QR code to the laptop computer, and the user scans this QR code by using his/her smartphone. Finally, the authenticator system grants or denies access to the laptop computer. Similarly to previous implementations, the same Android and web server application was used in the smartphone and the authenticator server. However, new client application have been designed and implemented for laptop computer.

After introducing particular implementations, the authentication protocol was verified by using an automated security protocol verification tool (Chapter 5). To check if the authentication protocol satisfied the authenticity and the secrecy properties, a security protocol verification tool needed to be used. At first, the authentication protocol was modeled using

a high level protocol specification language. Then, the protocol was verified to show the secrecy and the authenticity properties were not violated.

These implementations were tested and analyzed to demonstrate the performance variations over different versions of the protocol. The different implementations compared in terms of execution time, battery usage, network traffic and memory consumption. The most effective authentication performance was observed while using two smartphones and NFC technology for challenge transmission. Using the NFC protocol instead of QR code images significantly decreased the network traffic and battery usage. However, measured execution time results were close to each other.

The rest of the thesis is structured as follows: Chapter 2 expresses the related work. Chapter 3 describes the system design of the new authentication protocol. Chapter 4 introduces different real life implementations of authentication protocol. Chapter 5 analyzes the implementations in detail. Chapter 6 discusses future work, describes other possibilities, and concludes the thesis.

CHAPTER 2

RELATED WORK

Access control determines who can access system resources [7]. There are two main parts of broad access control definition, authorization and authentication [2]. Authorization is the process of specifying access rights to the resources. Further, authentication is the set of procedures that determines whether someone or something should be allowed access to some system or resources [2].

Authentication methods are a very common way to confirm the legitimacy of someone's or something's identity in digital and physical systems. The standard three factors in authentication techniques are something the user knows (e.g., a password), something the user has (e.g., a hardware token device) and something the user is (e.g., a fingerprint) [1]. In order to achieve an authentication, one of the factors can be used. Also, the factors can be combined to create a multi-factor authentication. This chapter compares and contrasts closely related common authentication methods with the new authentication protocol.

2.1 Authentication Using Something the User Knows

One of the most popular approaches is using something the user knows as an authentication factor. Passwords are an example of authentication methods based on something the user knows [1]. Furthermore, a username and password authentication scheme is mainly used in online web applications (e.g., online banking applications, social media platforms, mail clients).

Another approach to implement authentication mechanisms based on something the user know is asking personal questions from the users, such as "What was the make of your first car," "What is your first pet's name" [8]?

2.1.1 Advantages and Disadvantages

A major advantage of the traditional username and password scheme is that it is very easy to implement. A wide variety of password encryption tools exist in order to store passwords in the systems. Moreover, password authentication is relatively easy to use because users are accustomed to it [9].

Researchers have been studying the vulnerabilities of the username and password schemes. One of the common problems with passwords is attackers can guess the password [10]. Users are more likely to choose simple passwords in order to better remember them in the future. Spear phishing and social engineering is also another major attack model with passwords. The attacker can send a fake email and ask for the username and/or password from the user [11, 12]. Also, another survey showed that more than 70% of people would reveal their passwords for just a candy bar [13].

2.2 Authentication Using Something the User Is

Biometric identifiers, e.g., finger prints, voice prints, retina scans, are examples of an authentication scheme based on something the user is. Before the authentication starts, the authenticator system needs to take users' biometric measurements in order to identify them. Then, when users want to access the system, the authentication mechanism analyzes and verifies users' identities [1]. In order to identify users, systems can read fingerprints, scan retinas, scan voices, read signatures.

2.2.1 Advantages and Disadvantages

Since users can forget passwords and lose hardware devices (e.g., hardware token, bank card), something the user is based authentication mechanism could be used. Furthermore, these authentication schemes are cost-effective; after setting up the authentication infrastructure, there is no need for extra device utilization.

User identity theft is one of the major concerns with "something the user is" based authentication mechanisms. The attacker could steal the fingerprints of users and authenticate the system. Similarly, the attacker could also spoof voice samples, retina images or signatures of users [14, 15].

The replay attack is also another vulnerability in the biometric authentication scheme [14]. The attacker could steal the fingerprint information from the authenticator device after a successful authentication. Additionally, uniqueness is another concern for biometric authentication. Two different people could have the same characteristics of their faces, signatures or voice prints [16].

2.3 Authentication Using Something the User Has

Some authentication schemes require a physical object in order to complete authentication. These physical objects, known as "something the user has," are factors of the authentication. Physical tokens are one of the most popular examples of this scheme [17]. RSA securId [18], Battle.net authenticator [19], yubico [20] are instances of implementation of the physical tokens.

Magnetic strip cards are another popular example similar to hardware tokens. Magnetic strip cards (e.g., id cards, credit cards, smart card) are widely used to authenticate the users into the systems.

2.3.1 Advantages and Disadvantages

A wide variety of solutions has been introduced to identify the users with physical authenticator devices [21, 22, 23, 24]. The popularity of studies on the something the user has authentication schemes leads to cost-effective solutions to authentication problems.

Theft-based attacks is one of the major concerns in something the user has based authentication mechanisms [4]. A physical token device, such as an id card, or a credit card could be stolen by an attacker and the attacker could easily access systems. In addition, another common attack model on token devices are the replay attacks [25], in which, physical token devices or credit cards could be copied by an attacker and used to gain access to systems.

Clock drift, battery and synchronization problems are other examples of physical token issues. In a certain time period, users need to do maintenance in order to reuse their token devices [26].

2.4 Multi-factor Authentication

Multi-factor authentication mechanisms combine at least two of three authentication factors (something the user knows, something the user has, something the user is). Automated Teller Machines (ATMs) extensively uses the two-factor authentication scheme during the bank transactions. For example, withdrawing money from ATM requires a bank or a credit card (something the user has) and a personal identification number (PIN) (something the user knows). Another common example of multi-factor authentication schemes is a combination of an RSA securID physical token and a password [27].

2.4.1 Advantages and Disadvantages

In the single factor authentication schemes, a token device can be stolen or a password can get compromised. Introducing multiple authentication factors can diminish the attack surface. For example, if an attacker compromises the hardware token in a two-factor authentication scheme using a password and a hardware token, the system still remain inaccessible.

Although multi factor authentication schemes have improved the security compared to single factor authentication schemes, the usability is a major concern [28]. Users may need to carry additional devices (e.g., physical token) on them. Moreover, since there are multiple factors involved in authentication, extra steps need to be performed in order to complete authentication. Users can get tired of performing extra steps and disable the multi-factor authentication feature from systems.

2.5 Challenge-Response Mechanisms

Challenge-response mechanisms use one-time usable identifiers as an authentication factor. For example, instead of using the same password, the authenticator system uses one-time passwords [1]. One-time passwords are the ideal example of challenge response mechanisms. In every authentication attempt, the system generates a random password and sends it to the user. In addition to one-time passwords, using a hardware token (e.g., mobile phone, physical token device) is another approach to implement challenge-response mechanisms.

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is another example of a challenge-response mechanism that tests whether the user is human or not [29].

2.5.1 Advantages and Disadvantages

Challenge-response mechanisms are robust against bot attacks on the system [1]. Furthermore, many challenge-response mechanisms (e.g., CPATCHA) are very popular and easy to implement.

Dictionary attacks are a common attack model on challenge-response mechanisms [1]. If an attacker knows the challenge and the response, a dictionary attack can be performed to predict one-time passwords. Furthermore, there are many studies showing that CHAPCHA-based mechanisms can get compromised by automated systems [30, 31, 32].

CHAPTER 3

SYSTEM DESIGN

This chapter introduces the system design of the new authentication protocol. Unlike the traditional authentication techniques (e.g., username and password, bio-metric identity, RSA securId), this new authentication method requires at least two associated (paired) devices to complete the authentication process.

This new authentication method is a one-factor authentication mechanism. Furthermore, this technique uses at least two user devices as the "what the user has" factor of authentication. Instead of using additional devices (e.g., hardware token, id-card), using user devices is a cost-effective solution to implement this authentication protocol. Therefore, the authentication protocol implementations are easy to deploy into systems.

The association of multiple devices with a user is an innovative way to prevent device theft. All user devices need to be granted a private key or create their own private key, during the registration phase. In this regard, all of the associated devices need to participate in the authentication process. For example, if two smartphones are associated with a particular user and one of them is stolen, the attacker needs the private key of the other device to authenticate the stolen device.

The public key infrastructure (PKI) has been used to ensure confidentiality and integrity of shared information during the authentication process [2]. The device pairing process is an essential part to sharing public keys of the devices. The device pairing process should be done in registration phase and the registration should be performed before the authentication starts.

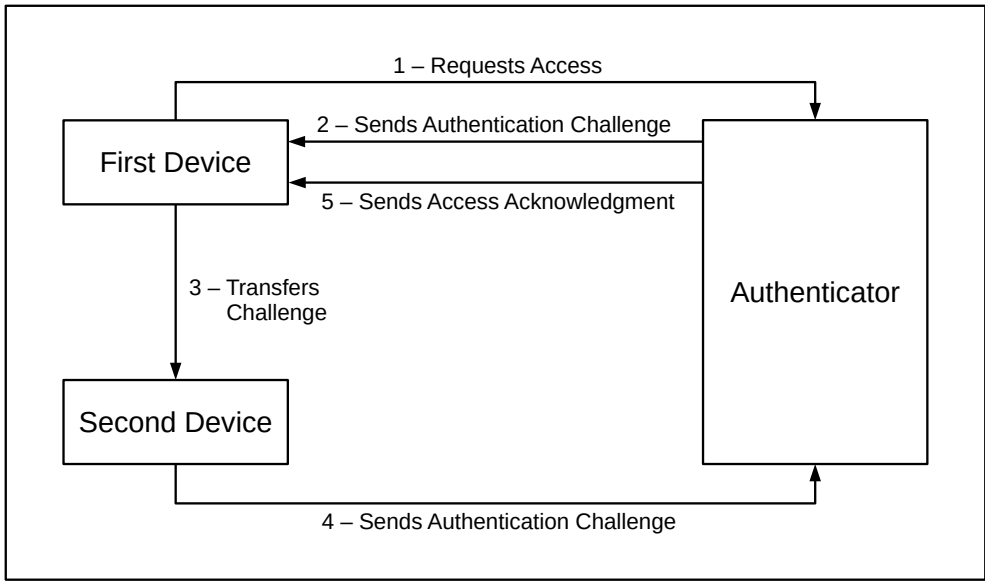

Figure 3.1. System design overview.

During the current investigation and analysis of the protocol, the following assumptions have been made:

- The public key infrastructure is sound and robust against network-based attacks (e.g.: man in the middle, eavesdropping, masquerading).
- The device registration process was completed properly and the devices were paired with each other and the user.

Figure 3.1 provides an overview of the authentication protocol with two devices. In the first step, the authentication process starts with an access request from the first device. Then in the second step, the authenticator server generates an authentication challenge and sends it to the first device. In the third step, the first device transmits the challenge to the second device. In the fourth step, the second device generates a response and sends this response to the authenticator server. Finally, in the fifth step, if the received authentication challenge is valid, the server sends a successful access acknowledgment to the access requesting device.

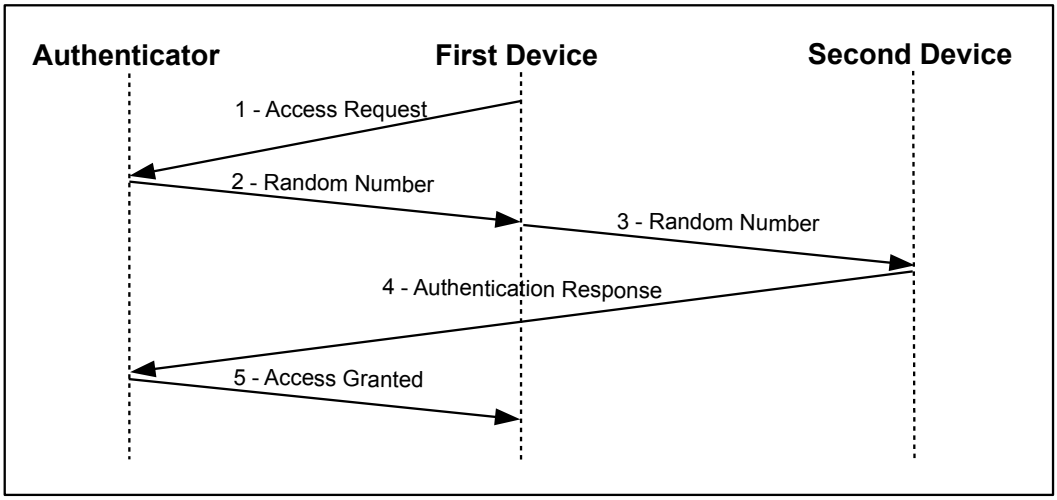

Figure 3.2. An example of the authentication protocol using two devices.

3.1 Authentication

The authentication process is the main contribution of the authentication protocol. However, in order to start the authentication process, the user needs to pair at least two devices and register them into the authenticator system. The device pairing process will be explained in Section 3.2 in detail.

3.1.1 Two-device Authentication

The authentication protocol requires at least two devices, which are associated with a user, to perform an authentication process. Figure 3.2 describes in detail an example of the authentication protocol using two devices. All the communication in the system was designed with public key infrastructure. The device and the server have a pair of public and private keys to encrypt communication. Additionally, all network messages contain a time-stamp and a digital signature. PKI, digital signatures and time-stamps are vital factors of the system design setup in order to prevent man in the middle, masquerading, and denial of service attacks [33].

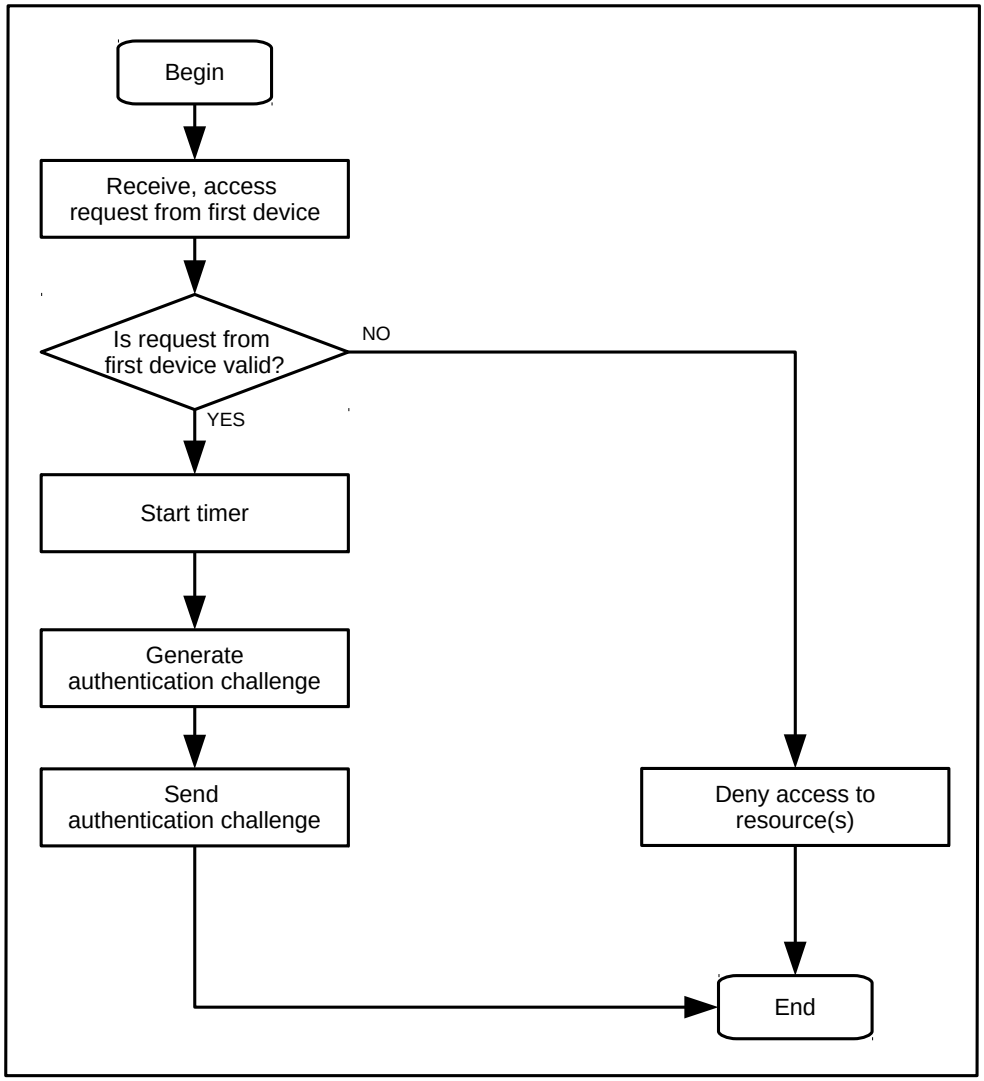

Figure 3.3. Access request handling algorithm.

In the first step, the first device sends an access request to the authenticator server. This encrypted access request message contains a user identification information (e.g., email address of user), a device identification information (e.g., device name), and a time stamp. Besides this information, the message contains a digital signature that signed with the private key of the first device.

After the authenticator server receives and encrypts this message, it runs an algorithm to validate the access requesting device (first device) as shown in Figure 3.3. At first, the authenticator server checks the first and the second devices' identification information.

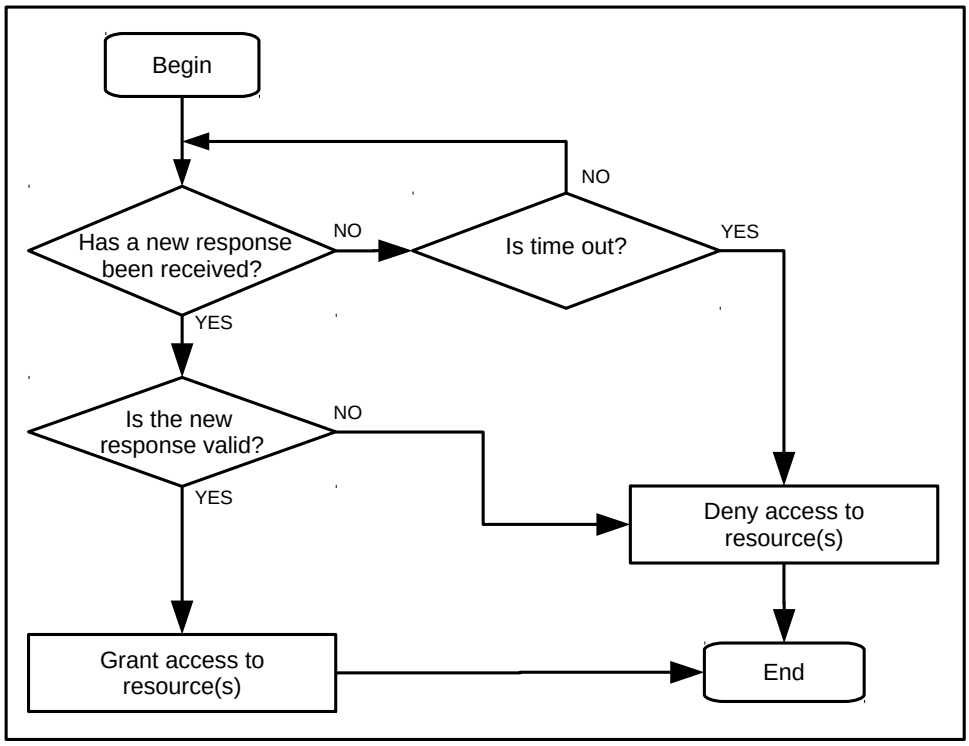

Figure 3.4. Challenge verification.

These two devices need to be associated with each other in the system database. The device association process is described in Section 3.2. Then, the server generates a random number, and it starts a timer to limit the authentication time.

In the second step, the authenticator server sends the random number to the first device. The randomly generated number is used as a challenge in the authentication protocol.

In the third step, the first device transmits the challenge to the second device. The challenge transmission could be done in many alternative ways. For example, images, QR codes, NFC protocol, vibration, sound and light waves, and infrared technology could be used.

In the fourth step, after the second device receives the challenge, it forwards the challenge to the authenticator server.

At this point, the authenticator server runs an algorithm to validate the challenge as specified in Figure 3.4. First, the timer checks if the authentication time is expired. Then,

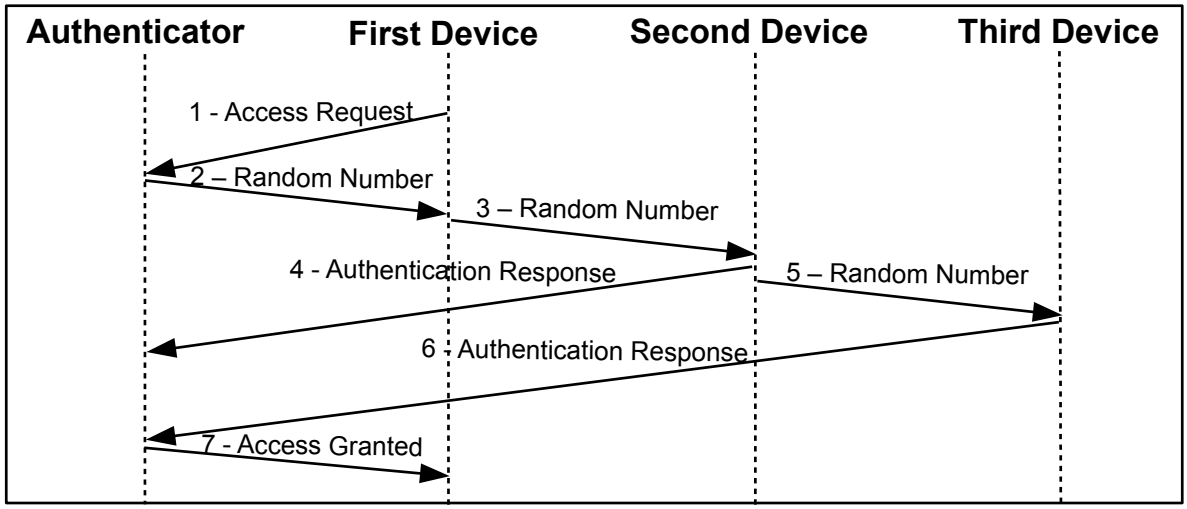

Figure 3.5. An example of the authentication protocol using three devices.

the authenticator server checks if the second device is paired with the first device. Finally, the authenticator server checks if the challenge is valid. If all these requirements are met, in the fifth step, the authenticator server grants access to the first device.

3.1.2 Three-device Authentication

Two-device authentication is not the only version of the authentication protocol. Hence, more than two devices could be used to implement this authentication process. Also, since an attacker needs to steal all associated devices in order to access the system, introducing more devices brought extra layers of security into the authentication process.

Figure 3.5 illustrates another example of the authentication protocol using three devices. The devices shown in Figure 3.5 need to be associated with the same user. In the first step, the first device requests access to the authenticator server. Then, in the second step, the authenticator server generates a random number and sends it to the first device. In the third step, the second device receives the random number from the first device. In the fourth step, the second device sends an authentication response back to the server. In the fifth step, the second device sends this random number to the last device. In the sixth step,

the third device then sends another authentication response back to the authenticator server. If the authenticator server determines that the responses received from the second and third devices are valid for the issued random number, the authenticator system grants the first device access to the resource.

3.2 Registration

Registration is a vital part of the authentication protocol. In order to start an authentication process, at least two devices need to be associated.

During the authentication process, the authenticator server needs to know two pieces of basic information: the identity of the paired devices, and the identity of the user who owns the devices. To collect this information, a unique device name could be used as an identifier of the device. Also, an email could be used to identify the user.

Although this proposed approach is used in this authentication protocol, it's not the only way to implement the registration process. Many existing techniques can be used to enforce the registration process. Registration with call center, short message service (SMS) activation or many alternative approaches could be used.

Device registration is not the main contribution of this thesis. Hence, the security aspects of the registration phase is not addressed in this thesis.

CHAPTER 4

IMPLEMENTATION

Implementing the authentication protocol has been useful for confirming its practicality. This section introduces the infrastructure (Section 4.1) and the cryptographic (Section 4.2) setup of the implementation. Then, the device pairing process is explained in Section 4.3. Finally, this chapter presents multiple implementations of the authentication protocol with different system configurations (Section 4.4).

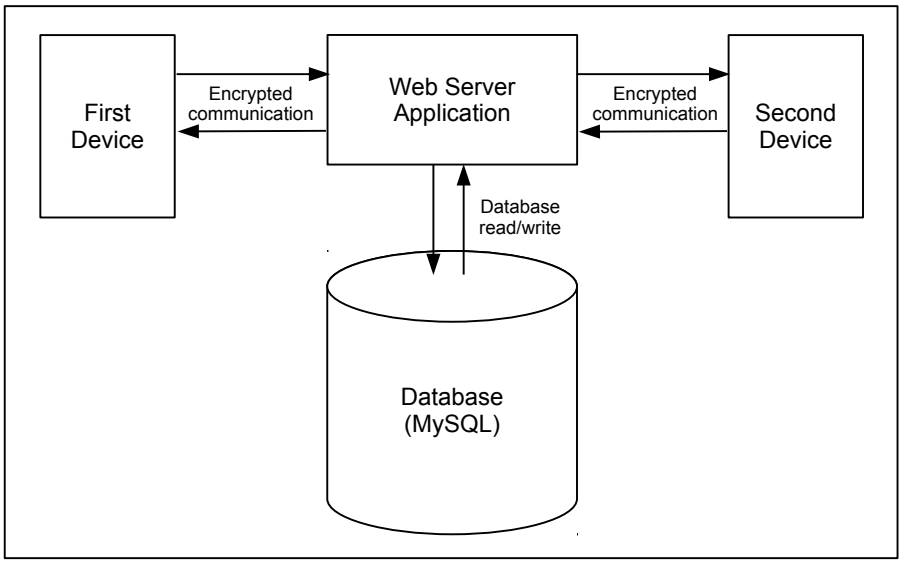

Figure 4.1. Technical infrastructure design.

4.1 Infrastructure Setup

Implementing different versions of the authentication protocol requires flexible software architecture in order to reduce development overhead. Hence, a flexible server infrastructure

was designed and implemented. Thanks to this infrastructure design, the authenticator server implementation does not need to be changed while implementing new versions of the authentication protocol. Furthermore, the rest of the different authentication protocol versions were developed based on this particular infrastructure setup.

4.1.1 Server Architecture

The principal mechanism to implement this authentication protocol is the authenticator server. A web server application is suitable to implement the authenticator system of the protocol as shown in Figure 4.1. The server application communicates with devices via REST API [34]. Representational State Transfer (REST) is a software design approach to implement web services. Thus, this design approach allows the web server application to communicate with different devices regardless of their operating systems.

The web server application was implemented by using Spring Framework [35] in Java language. The Spring Framework is a Java-based tool for developing and configuring the enterprise web applications.

4.1.2 Database Design

Another essential contribution of this protocol is the device association process. To associate devices, a device and user identification information are required. Hence, a simple database was designed and implemented as shown in Figure 4.2. An email address was used to identify the users. Also, the unique device name was used to identify the associated devices. The associated device identity information (device name and email address) was created during the registration phase and stored in the database.

For data storage and data persistence, MySql database and Hibernate framework [36] were used. Hibernate is an open source object-relational mapping library project for Java language. The goal of using a third party framework to manage database operations is to prevent SQL injection attacks [37].

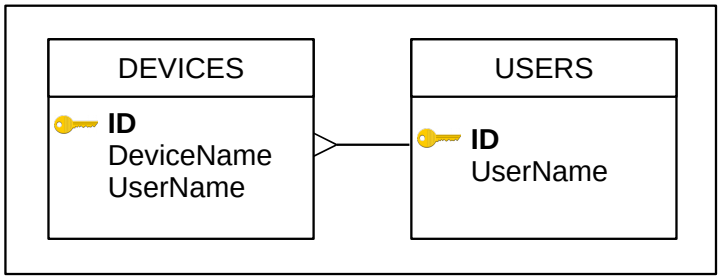

Figure 4.2. Database design.

4.1.3 Software Management and Deployment

For build automation and software dependency management, Apache Maven was used. Apache Maven is a software project management tool for software projects [38].

This server application was deployed on the Apache Tomcat [39] application server. Apache Tomcat is an open source Java servlet implementation for web applications.

4.2 Public Key Infrastructure Setup

The asymmetric encryption was implemented for all communication steps in the authentication protocol. To implement asymmetric encryption, RSA encryption algorithm was used in Java [40]. During the registration process, the server shares its public key with the device. Then, the device creates its own public and private key pair and sends the public key to the server. In every communication step, the server encrypts the data with devices' public key. Also, the devices decrypt the data with their private key as shown in Figure 4.3. For example, in the second step in Figure 4.3, the server generates a random number and sends it to the first device. The server encrypts this message with the first device's public key. After the first device receives this encrypted data, it decrypts by using its private key.

In order to prove the authenticity of the data in the system communication, a digital signature scheme was used. In every communication step, the data are signed with the devices' private key as shown Figure 4.3. First, the server creates a hash function by using SHA-1 hashing algorithm. SHA-1 is a cryptographic hash function [41]. Then, in the second

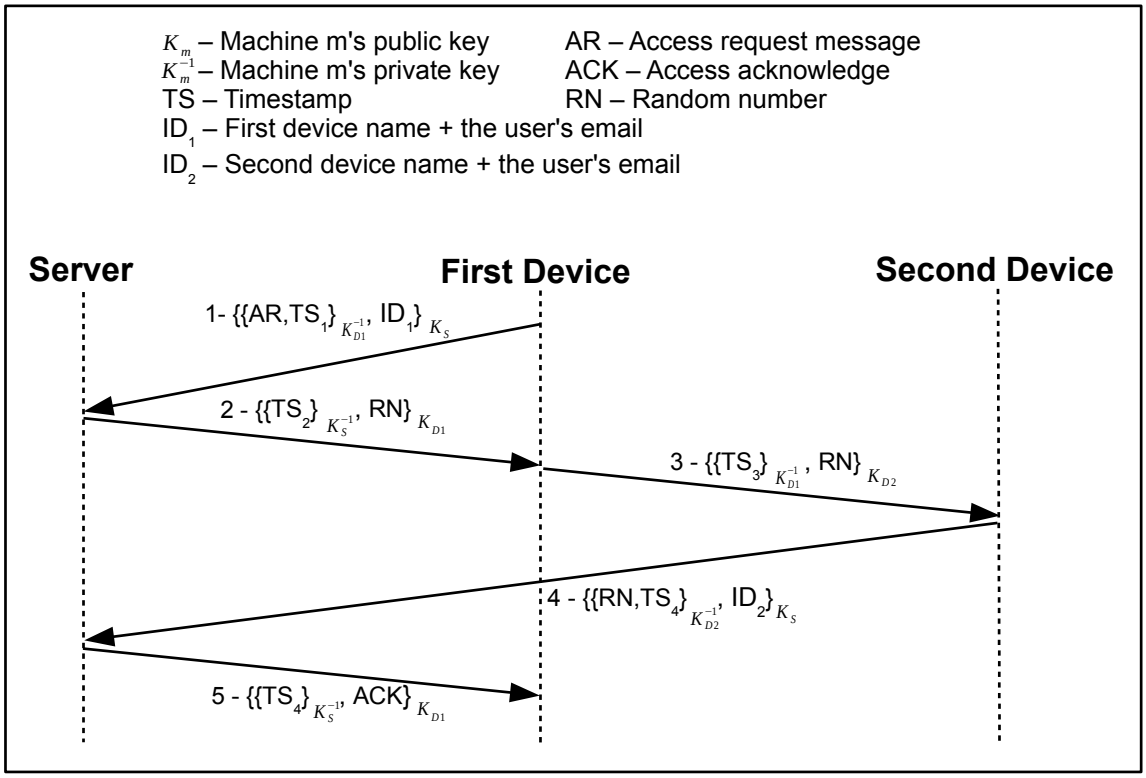

Figure 4.3. Implementation of the system design.

step, the server encrypts the hash function with its own private key by using RSA encryption algorithm, which creates the signature. Finally, the signature and the data are merged by the server and sent to the device as digitally signed data. After the device receives the signed data, it needs to verify it. First, the signature is decrypted with the server's public key by using the same algorithm; then the device creates another hash code from the data. Finally, if the two hash codes match with each other, the authenticity of the data is ensured.

4.3 Registration Phase

The authentication protocol and registration methodology were discussed in Chapter 3. One particular registration process was implemented in order to perform the authentication protocol. In this section, the registration phase implementation is explained in detail.

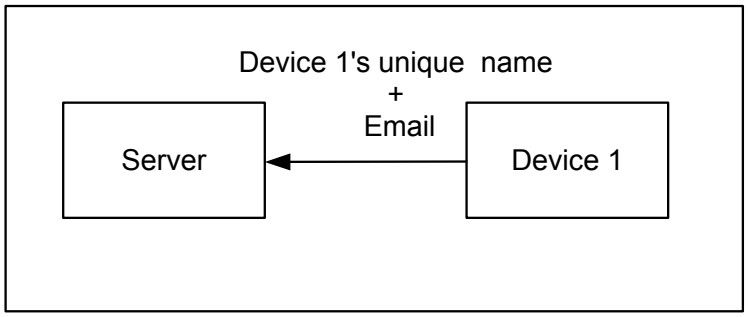

Figure 4.4. First step of the device pairing process.

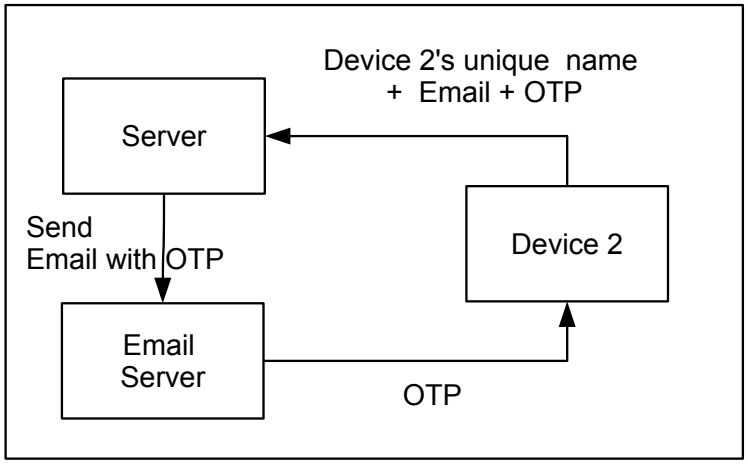

Figure 4.5. Second step of the device pairing process.

Figure 4.4 and Figure 4.5 demonstrate the registration process in two steps. In the first step (Figure 4.4), an email address and a unique device name need to be entered to identify the device and the user. After the server receives the registration request from the first device, first it validates the information by checking whether the device name is unique and the email address has not been registered before. If the device and the user are valid, the server starts a timer to open a time frame to limit second device registration duration. If the second device isn't registered in this time frame and the timer expires, the email becomes blacklisted in the server.

In the second step (Figure 4.5), the second device sends the same email address and a unique device name. Then, the server validates the user and the device identities by applying

the same process in the first step. In this moment, in order to confirm the user's identity, a One Time Password (OTP) was used. The server generates an OTP by using Java security API [42]. Then, the server sends the OTP to the email address, which is already provided by the user. The OTP should be entered on the second device within the time frame, which is already started in the first step by the server. After this step, if the timer is still not expired, the server associates these two devices with each other and with the user. Furthermore, the server stores the identity information of the associated devices in the system database.

4.4 Authentication Phase

In this section, a different set of implementations of the authentication process will be presented. All these implementations were embodied from the authentication protocol concept presented in Chapter 3. Although these implementations were designed with a different set of system configurations, they use the same shared modules.

The following implementations use the same infrastructure implementation presented in Section 4.1. As an authenticator system, the web server implementation presented in Subsection 4.1.1 was used. For the device association and the user initialization, the registration phase presented in Section 4.3 was used. Furthermore, the cryptographic infrastructure implementation presented in Section 4.2 was used to ensure the network security of the implementations.

4.4.1 Implementation: Device One: a Smartphone, Device Two: a Smartphone, Challenge Transmission by QR Code

In this implementation, two Android smartphones were used to represent the first and second devices in the authentication protocol. Additionally, an Android application was designed and implemented to operate the authentication process. Before the authentication process starts, the registration phase presented in Section 4.3 was performed in order to associate the devices with each other.

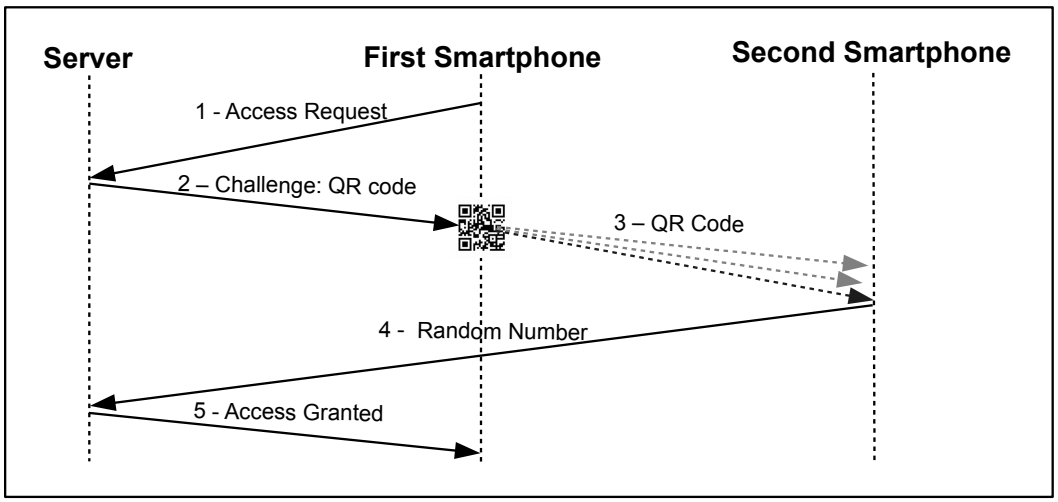

Figure 4.6. Implementation: two smartphones - QR code.

Figure 4.6 demonstrates this specific implementation. After the first step, when the server receives the access request, it creates a 32-bit alpha numeric random number using Java security API [42]. Then, the server generates a QR code from the random number. To compute a QR code from the random generated number, Google Zxing open source framework [43] was used.

In the second step, the server sends this QR code as a challenge. After the second step, when the first smartphone receives the challenge, it locally broadcasts the QR code to the other associated device by displaying the QR code on its screen as shown in Figure 4.7. The second device scans the QR code (Figure 4.8) and extracts the random number, which was already created by the server. To read the QR code, Google Zxing framework was implemented in the Android application. In the fourth step, the second smartphone sends the random number to the server. The server runs an algorithm to validate the request as discussed in Chapter 3. In the fifth step, if the request is valid, the server grants access to the smartphone. Finally, the server successfully authenticates the Android application.

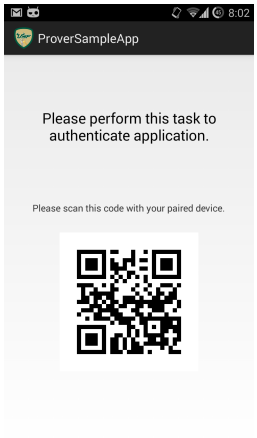

Figure 4.7. QR code displayed.

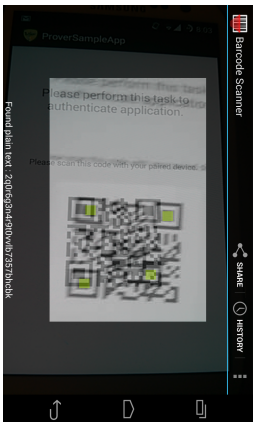

Figure 4.8. Scanning the QR code.

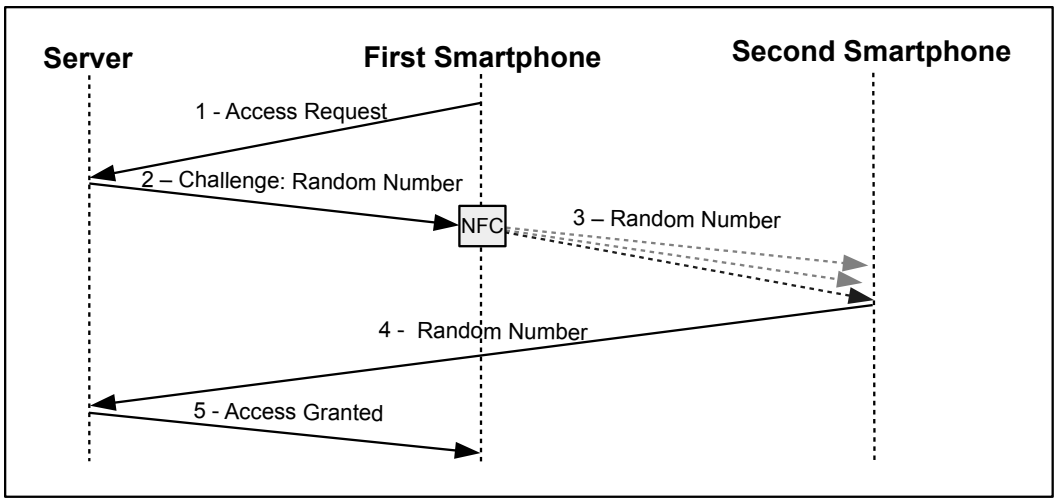

Figure 4.9. Implementation: two smartphones - NFC.

4.4.2 Implementation: Device One: a Smartphone, Device Two: a Smartphone, Challenge Transmission by NFC

Similarly to the implementation presented in Subsection 4.4.1, this implementation uses two Android smartphones during the authentication process. However, for the challenge transmission, Near Field Communication (NFC) technology was implemented. Instead of scanning the QR code, two smartphones need to be tapped together in order to transmit the challenge.

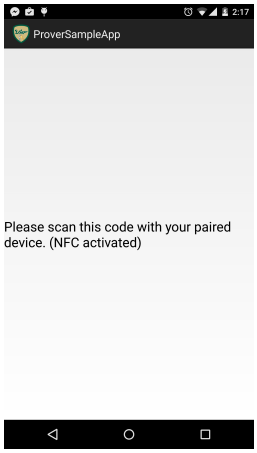

Figure 4.10. Challenge locally broadcasted.

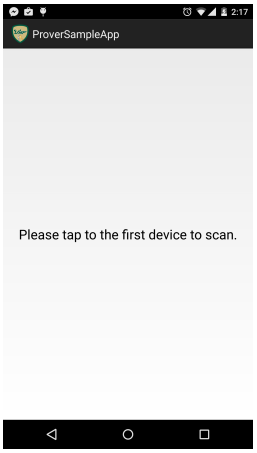

Figure 4.11. Extracting the number.

Figure 4.9 illustrates this specific implementation with NFC. Different from the previous implementation (Subsection 4.4.1), after the first step, when the server receives the access request, the server creates a random number, but it doesn't generate a QR code. Instead of sending the QR code, the server sends the generated random number to the first device as shown in the second step. When the second device receives the random number, it locally broadcasts it by opening an NFC connection as shown in Figure 4.10. NFC is a set of technologies that transmits data over radio waves by bringing the devices into proximity (generally a distance of 10 cm or less) [44]. To implement the challenge transmission process via NFC, Android NFC library was used [45]. In the third step, to perform the challenge, users need to tap devices together to establish radio communication as shown in Figure 4.11. Then, the second smartphone extracts the random number and sends it to the server as shown in the fourth step. Finally, the server validates the request and grants access to the first device.

4.4.3 Implementation: Device One: a Laptop Computer, Device Two: a Smartphone, Challenge Transmission by QR Code

In this implementation, a laptop computer represents the first device and a smartphone represents the second device. For challenge transmission, a QR code was implemented.

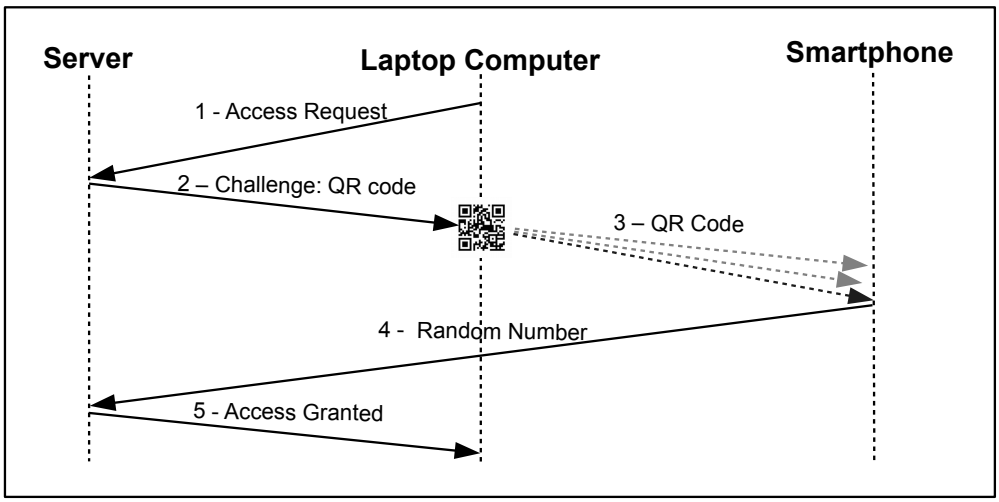

Figure 4.12. Implementation: laptop - smartphone - QR code.

To associate the smartphone with the laptop computer, the particular registration phase implementation was performed (Section 4.3). An Android application, which is the same application used in the first and second configurations, was used in this implementation to operate authentication process on the smartphone. For the laptop computer (the first device) a Java desktop application was designed and developed. For this implementation, a laptop computer was used. Additionally, the developed desktop application was designed to be operated on cross platforms. Hence, it can run on Windows, Mac OS X or Linux empowered computers.

Figure 4.12 demonstrates this specific implementation in detail. In the first step, the laptop computer wants to access to the system. Similar to the first implementation as presented in Subsection 4.4.1, the server sends a challenge via QR code. In the second step, the computer receives this QR code and locally broadcasts it by displaying it on the screen as shown in Figure 4.13. And the rest of the process follows as similar to the first implementation. The smartphone scans the QR code, extracts the random generated number and sends it to the server. Finally, the server grants or denies access to the laptop computer. This implementation allows users to access the systems from their laptop computer by using their smartphone.

Figure 4.13. QR code displayed on the laptop screen.

CHAPTER 5

EXPERIMENTAL RESULTS

In this chapter, the new authentication protocol design presented in Chapter 3 will be examined, and the implementations presented in Chapter 4 will be tested in detail. First, the authentication protocol was validated by using an automated validation tool for Internet security protocols. Then, the implementations were tested by using different devices and platforms to determine their performance.

5.1 Model Checking

In this section, the authentication protocol is modelled and verified. To verify that the authentication protocol guarantees authenticity and secrecy of the communication between devices and the authenticator server, a model checker must be used. Thus, AVISPA [46] model checker was used to check authenticity and secrecy properties. Automated Validation of Internet Security Protocols and Applications (AVISPA) is an automated model checker for large scaled security protocols. The AVISPA automation tool supports four verification backend tools [47]:

- OFMC (On-the-fly model checker)
- CL-AtSe (Constraint Logic based Attack Searcher)
- SATMC (SAT-based Model-Checker)
- TA4SP (Tree based model checker)

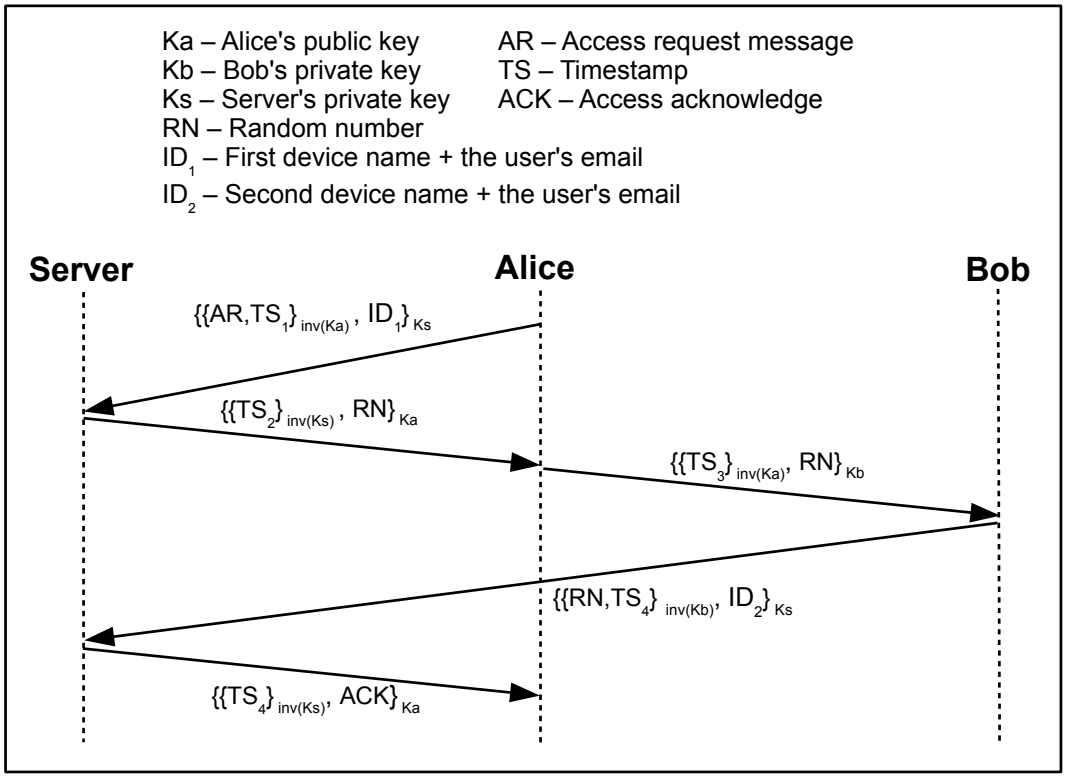

Figure 5.1. HLPSL specification in Alice and Bob notation.

CL-AtSe and SATMC backend platforms are used to verify the bounded number of sessions and protocol falsification. OFMC backend is useful for detecting guessing and replay attacks [47]. TA4SP backend provides unbounded security protocol verification by using tree-based languages [47].

5.1.1 Modelling the Protocol

In order to verify the authentication protocol with AVISPA, the protocol was modelled with HLPSL. High-Level Protocol Specification Language (HLPSL) is a language for modelling and specifying security protocols. HLPSL uses Alice and Bob notation to model security protocols [48].

The authentication protocol was modelled by using HLPSL in Alice and Bob notation as shown in Figure 5.1. The server represents the system that authenticates the devices.

Alice represents the first device and Bob represents the second device of the authentication protocol. Ka, Kb, Ks are the public keys of the first device, the second device, and the server respectively. The inverse function (inv) retrieves the private keys of the public keys.

There are two security goals exist in the AVISPA. To verify if the devices are being authenticated, the following goals were specified:

- Authentication on the first device
- Authentication on device two

Furthermore, to verify if the authentication communication was kept secret, the following goal was specified:

- Secrecy of hashed message

5.1.2 Results

The automated validation was performed on the specified protocol (Figure 5.1) by using the AVISPA model checker. During the experiments a laptop computer was used. The laptop computer is a MacBook Pro, which has 8GB RAM, a 2.2 GHz Intel Core i7 processor and 750GB storage.

Table 5.1 summarises the results of the verification. To verify the bounded number of sessions and protocol falsification, CL-AtSe and SATMC backend platforms were used [47]. CL-AtSe completed the verification in 6 seconds by analyzing 2470 states. Also, it took 285.13 seconds to complete verification with SATMC. As a result, both backends did not find any possible attacks on the protocol. To detect guessing and replay attacks, the protocol was verified by OFMC backend. OFMC ran a heuristic search algorithm with 1000000 plies and analyzed 68 total nodes. As a result, the authentication protocol proposed in Chapter 3 was validated by using an automated tool for verification of the security protocols.

Table 5.1. Model checker result

Backend	Summary	Statistics
CL-AtSe	SAFE	Analysed: 2470 states Reachable: 2470 states Translation: 0.02 seconds Computation: 6.0 seconds
SATMC	SAFE	attackFound: false upperBoundReached:true graphLeveledOff: 7 steps satSolver: zchaff solver maxStepsNumber: 30 steps stepsNumber: 7 steps encodingTime: 285.13 seconds solvingTime: 0 seconds if2sateCompilationTime: 0.49 seconds
OFMC	SAFE	parseTime: 0.00s searchTime: 5.86s visitedNodes: 68 nodes depth: 1000000 plies

5.2 Experimental Testing

Different versions of the authentication protocol were designed and implemented as presented in Chapter 4. In this section, the presented implementations will be explored to measure the performance overhead.

5.2.1 Experimental Setup

The implementations of the authentication protocol use two devices and a web server application. During the experiments, three different Android smartphones and two different laptop computers were used. Table 5.2 shows the technical specifications of the devices, which are used during the experiments. One device is a Samsung Galaxy S3 I9300, which has a Quad-core 1.4 GHz processor, 1GB RAM, 16GB storage, a 2100mAh lithium ion battery, a 8MP camera and Android 4.4.4 operating system. The second mobile device is a LG Nexus

Table 5.2. Test devices' technical specifications

Device name	**Technical specification**
Samsung Galaxy S3 I9300	Quad-core 1.4 GHz CPU 1GB RAM 16GB storage 2100mAh lithium ion battery 8MP camera Android 4.4.4 operating system
LG Nexus 4	Quad-core 1.5 GHz CPU 2GB RAM 16GB storage 2100mAh lithium ion battery 8MP camera Android 4.4.4 operating system
Motorola Nexus 6	Quad-core 2.7 GHz CPU 3GB RAM 32GB storage 3320mAh lithium ion battery 13MP camera Android 5.0 operating system
MacBook Pro	2.2 GHz Intel Core i7 CPU 8GB RAM 750GB storage Mac OS X Yosemite operating system
Windows computer	2.1 GHz Intel Core i5 CPU 8GB RAM 250GB storage Windows 7 operating system

4, which has a quad-core 1.5 GHz processor, 2GB RAM, 16GB storage, a 2100mAh lithium ion battery, a 8MP Camera and Android 4.4.4 operating system. The last smartphone is a Motorola Nexus 6, which has Qualcomm Snapdragon 805 processor with 2.7GHz quad-core processor, 3GB RAM, 32GB storage, a 3320mAh lithium ion battery, a 13MP camera and android 5.0 operating system. The web server application was run on a MacBook Pro, which has 8GB RAM, a 2.2 GHz Intel Core i7 processor and 750GB storage. The other laptop computer is a Windows 7 machine, which has 8GB RAM, a 2.1 GHz Intel Core i5 processor and 250GB storage.

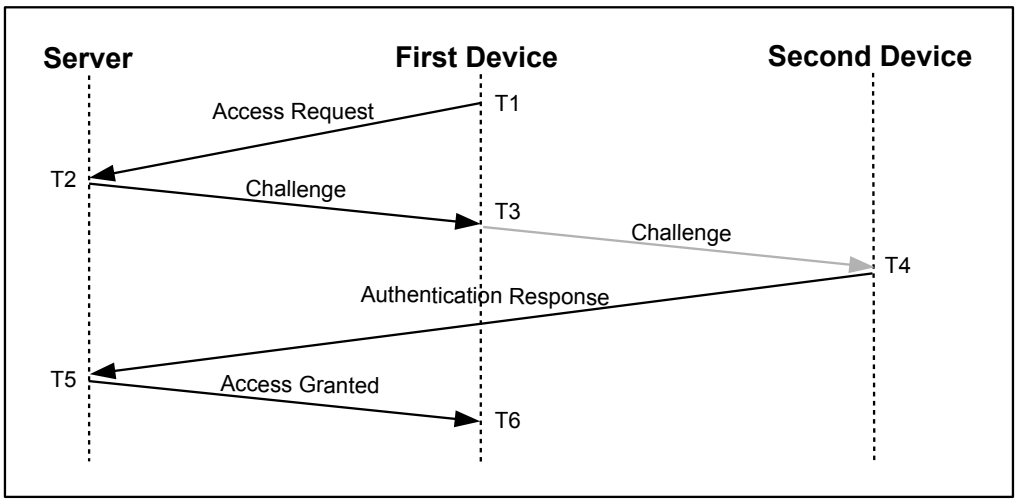

Figure 5.2. Authentication time measurement of the protocol.

For each protocol implementation, the same tests were performed. For each test, 20 iterations of the authentication procedure were executed to measure the average authentication time, memory usage, network consumption, and battery drain on each device. Authentication time was measured by summing up two different timer functions not including human interaction time. The first timer starts at time T1 and ends at time T3 as shown in Figure 5.2. Then the second timer starts at time T4 and ends at time T6. As a result, execution time calculated as shown below:

$$ExecutionTime = (T3 - T1) + (T6 - T4)$$

The memory usage was measured by Android DDMS (Dalvik Debug Monitor Server) [49]. The sampling frequency of the allocated memory on the heap was set to 10 seconds. The network consumption was measured by the network traffic tool provided by Android DDMS. To measure battery consumption, a third-party tool was used. During the experiments, mobile devices and computers never went to sleep. Target programs were compiled with Eclipse Luna using required frameworks. The web server application implementation has a total of 2184 lines of code (not including empty lines). The desktop application has a total of 1339 lines of code (not including empty lines) and the Android mobile application has a total of 2467 lines of code without empty lines.

Table 5.3. Protocol implementation test configurations

	Test configuration described in Subsection 5.2.2.1	**Test configuration described in Subsection 5.2.2.1**	**Test configuration described in Subsection 5.2.2.2**	**Test configuration described in Subsection 5.2.2.3**
First Device	Samsung Galaxy S3	LG Nexus 4	Samsung Galaxy S3	Laptop
Second Device	LG Nexus 4	Samsung Galaxy S3	LG Nexus 4	LG Nexus 6
Challenge	QR Code	QR Code	NFC	QR Code

Table 5.4. Experimental results with the first configuration described in Subsection 5.2.2.1

Device	**Execution Time (s)**	**Network Usage (bytes)**	**Memory Usage (MB)**	**Battery Consumption (Joule)**
First Device (Samsung S3)	0.0754	4749.3	4.055	14.4
Second Device (LG Nexus 4)	0.0754	476.2	9.380	4.1

5.2.2 Results

Table 5.3 shows the different test configurations with multiple devices. These test configurations were designed to evaluate the individual performance of each protocol implementation presented in Chapter 4.

5.2.2.1 Test: Device One: a Smartphone, Device Two: a Smartphone, Challenge Transmission by QR Code

The authentication protocol implementation presented in Subsection 4.4.1 was tested. Two Android smartphones were used to represent the first and second devices. In this implementation, two sets of tests were performed. First, the experiments were performed with a Samsung Galaxy S3 as the first device and an LG Nexus 4 as the second device. Then, the second experiments were performed with an LG Nexus 4 as the first device and a Samsung Galaxy S3 as the second device.

Table 5.4 shows the test results with a Samsung Galaxy S3 as the first device and an LG Nexus 4 as the second device. Similarly, Table 5.5 shows the results with an LG Nexus 4 as the first device and a Samsung Galaxy S3 as the second device. Average execution time for one complete authentication is 0.0754 seconds with the first configuration. To demonstrate the efficiency of the system, the time performance was measured with a different set of

Table 5.5. Experimental results with the second configuration described in Subsection 5.2.2.1

Device	**Execution Time (s)**	**Network Usage (bytes)**	**Memory Usage (MB)**	**Battery Consumption (Joule)**
First Device (LG Nexus 4)	0.0739	4674.3	10.2	11.3
Second Device (Samsung S3)	0.0739	481.2	3.3	5

smartphones, in which the first device was upgraded to a device with better resources (e.g., faster CPU, bigger memory) as shown in Table 5.5. As a result, execution time was not significantly affected by the specification of the device.

The network usage for the first device is approximately 10 times higher than the second device on each test. Indeed, the server exchanges more data with the first device. There are three network communications between the server and the first device and one of them contains an image. Between the server and the second device, there is only one network communication, and this communication does not contain large data (e.g., an image).

The LG Nexus 4 consumes more memory than the Samsung Galaxy S3 in each test. Memory allocation is managed by the Android operating system [50]. Since the LG Nexus 4 has larger memory, the memory allocation on runtime is larger for the LG Nexus 4 regardless of its role (e.g., the first or second device) in the authentication system.

Since both smartphones are using the same battery, it is expected that they will have close battery consumption results during the authentication process. The slight difference comes from the screen contrast setup of each phone. Furthermore, the network consumption causes a slight difference between the first and second device on the battery consumption. Since the first device uses more network, it also consumes more battery. To better understand the battery drain, average battery consumption of the Android message application was measured while sending a message. To send a message, this application consumes 7.4j energy with the Samsung Galaxy S3.

Table 5.6. Experimental results described in Subsection 5.2.2.2

Device	Execution Time (s)	Network Usage (bytes)	Memory Usage (MB)	Battery Consumption (Joule)
First Device (Samsung S3)	0.0689	628.2	3.65	8.2
Second Device (LG Nexus 4)	0.0689	476.2	8.992	5.6

5.2.2.2 Test: Device One: a Smartphone, Device Two: a Smartphone, Challenge Transmission by NFC

The authentication protocol implementation presented in Subsection 4.4.2 was tested. Similar to the previous implementation, this implementation also contains two Android smartphones to represent the first and second devices. Furthermore, NFC technology was used to transmit the challenge. One set of tests was performed to measure the performance differences while transmitting the challenge with NFC technology. The experiment was performed with a Samsung Galaxy S3 as the first device and an LG Nexus 6 as the second device.

Table 5.6 shows the experimental results for the specific implementation presented in Subsection 4.4.2. The average execution time for one authentication process is 0.0689 seconds. Since only one device is being authenticated during the authentication process, the same execution time was measured for both devices.

The network usage for the first device is approximately 30% greater than the second device. Sending a random number instead of a QR code image for challenge transmission results closer network usage numbers for both devices. Because the first device makes more network communications than the second device during the authentication process, the battery usage is slightly higher in the first device.

5.2.2.3 Test: Device One: a Laptop Computer, Device Two: a Smartphone, Challenge Transmission by QR Code

The last experiment was performed on the implementation presented in Subsection 4.4.3. Instead of using two Android devices, a Windows laptop computer and an Android smart-

Table 5.7. Experimental results described in Subsection 5.2.2.3

Device	**Execution Time (s)**	**Network Usage (bytes)**	**Memory Usage (MB)**	**Battery Consumption (Joule)**
First Device (Windows Computer)	0.0717	4987	80	16.9
Second Device (Motorola Nexus 6)	0.0717	520	22	6.1

Table 5.8. Implementations with different system configurations described in Chapter 4

	Implementation system configuration 1	**Implementation system configuration 2**	**Implementation system configuration 3**
First Device	Smartphone	Smartphone	Laptop
Second Device	Smartphone	Smartphone	Smartphone
Challenge	QR Code	NFC	QR Code

phone (Motorola Nexus 6) were used for the authentication process. For challenge transmission, a QR code image was used.

Table 5.7 presents the test results for the specific implementation presented in Subsection 4.4.3. Similar to the previous tests, the execution time measured the same for the first and second device. QR code image dominates the network usage on the first device. On the other hand, since the smartphone only sends a random number, not the QR code, the network usage on the second device is noticeably lower than the first device.

The first device uses 80 MB and the second device uses 22 MB of memory space. Memory allocation is entirely managed by the Windows and Android operating systems for the laptop computer and the smartphone respectively. Hence, the monitored memory space is not related to application size.

5.2.3 Analysis

The individual performance overheads for each implementation were presented in Section 5.2.2. In this subsection, performance differences between the implementations will be compared and analyzed with each other in detail.

Table 5.8 summarizes the hardware configurations used in the different implementations. The first and second implementation configurations use the same set of devices. However, for the challenge transmission, QR code was used in the first configuration and NFC technology

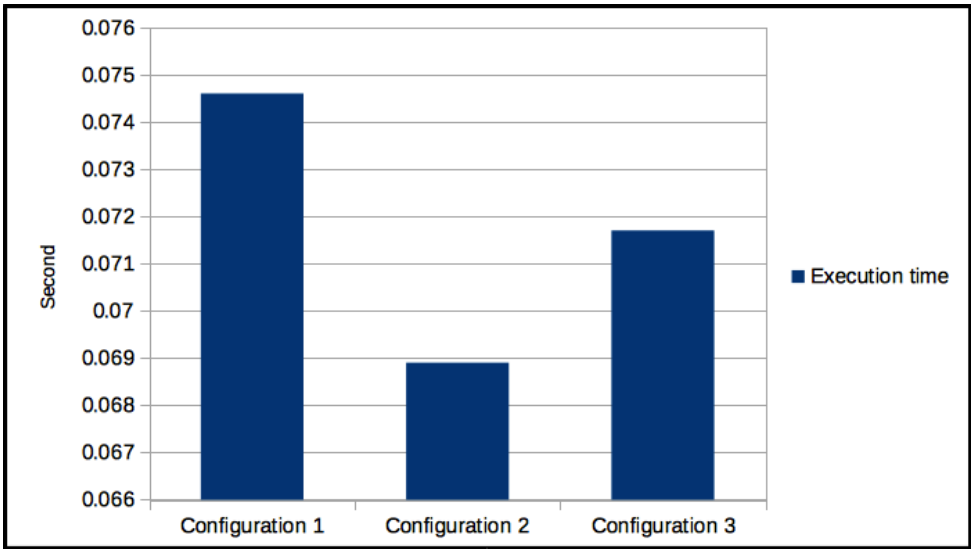

Figure 5.3. Average execution time for each configuration.

was used in the second configuration. The devices used in the third configuration have better system resources (faster CPU, larger memory) than the devices used in the first and second configurations.

5.2.3.1 Execution Time

Figure 5.3 shows the average authentication time for each implementation configuration. There is a slight performance difference between the first and second configurations. The only hardware difference between the first and second configurations is the first device. Instead of a smartphone, a laptop computer was used to represent the first device in the third configuration. For the challenge transmission both implementations use a QR code. Since the network card on the computer is faster than smartphones, the hardware of the devices cause that small performance difference.

A larger difference was observed between the first and second configurations. The authentication process is approximately 0.01 seconds faster in the second configuration. Although both tests were performed with the same devices, the second configuration uses NFC technology instead of a QR code. In the first configuration, after the server generates a random number, the server needs to generate a QR code from the random number. Whereas, in

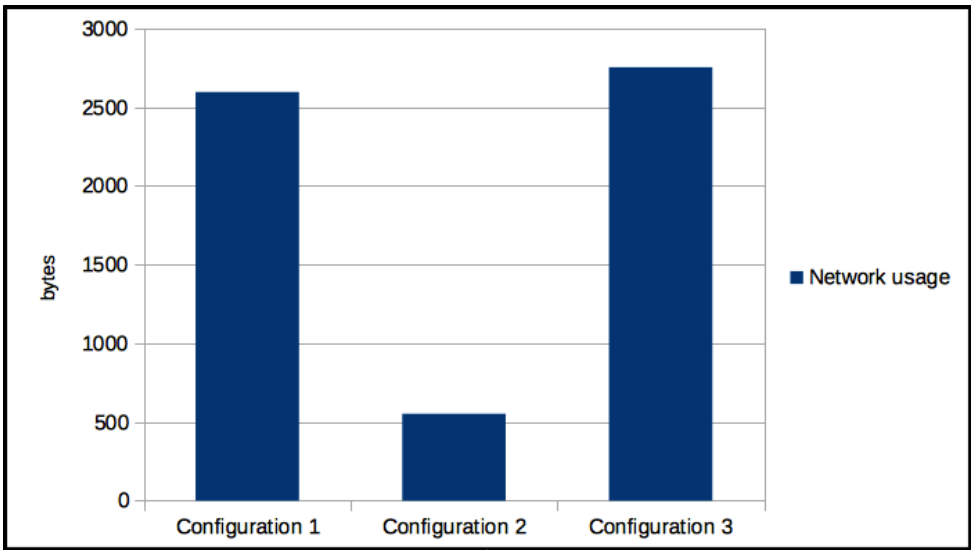

Figure 5.4. Average network usage for each configuration.

the second configuration, after the server generates a random number, the server directly sends the random number to the access requesting device. The QR code generation and transmission process adds around 0.01 seconds to the whole execution time.

5.2.3.2 Network Usage

Figure 5.4 illustrates the average network usage during the experiments. Similar to the execution time, the network usage results for the first and third configurations are fairly close to each other. Since they both use a QR code image for the challenge transmission, they exchange around 2.5MB of data during the authentication process.

The network usage for the second configuration is significantly lower than the network usage in the first and third configurations. Since the second configuration does not send or receive a large file (e.g., QR code image) during the authentication, the whole authentication can be performed using under 1MB of data.

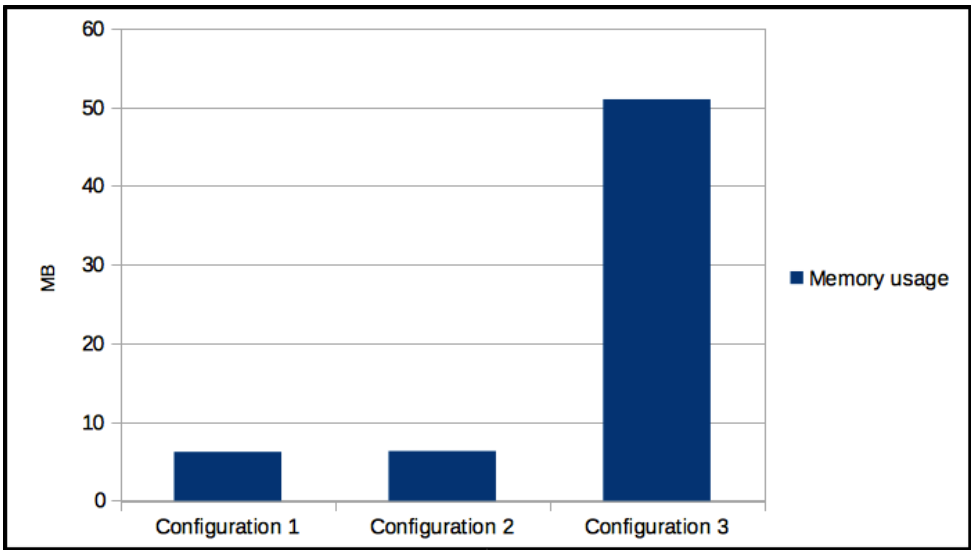

Figure 5.5. Average memory usage for each configuration.

5.2.3.3 Memory Usage

The dynamic memory allocation is mainly managed by operating systems. Hence, the developed applications for authentication protocol implementation do not have direct access to the memory. Although, to demonstrate the memory usage over different devices, memory tests were performed on each configuration as shown in Figure 5.5. The first and second configurations use the same set of devices (Samsung S3, LG Nexus 4). Therefore, these memory usage results are almost the same.

The third configuration was tested with two high-end devices. To represent the first device, a laptop computer with 8GB main memory was used. Also, to perform the challenge, a powerful smartphone (Motorola Nexus 6) with 3GB main memory was used. Since these devices have very large main memory compared to the Samsung Galaxy S3 and the LG Nexus 5, larger memory space was allocated for both applications by their operating systems.

Memory allocation results showed that the presented authentication method can run on devices, which have low memory capacities (less than 1GB main memory).

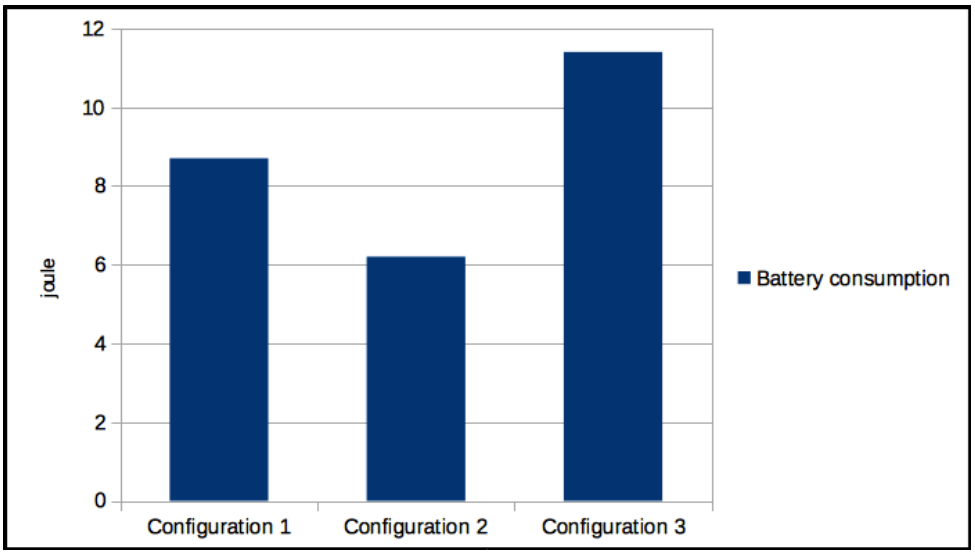

Figure 5.6. Average battery consumption for each configuration.

5.2.3.4 Battery Consumption

The average battery consumption is illustrated for each configuration in Figure 5.6. To better understand the battery consumption in different systems, Android messaging application's battery usage was measured while sending a message over the network. To send only one message over the network, the messaging application uses 7.4j of energy.

The second configuration consumes the least battery of all the configurations. Since the only difference between the first and second configurations was the challenge transmission, reading and decoding the QR code causes relatively higher battery consumption on devices. However, the largest battery consumption was noticed in the third configuration, which also uses a QR code for the challenge transmission.

The laptop computer and the smartphone used in the third configuration have more powerful CPUs than the devices used in the first configuration. Thus, more energy was consumed while performing an authentication process with a laptop computer and a more powerful smartphone. Nevertheless, devices used in the third configuration have larger batteries, thus the authentication process' impact on the battery usage was as low as the usage of the first and second configurations.

5.2.3.5 Conclusion

The authentication protocol implementations' performances were analyzed in terms of execution time, network usage, memory usage, and battery consumption in the previous subsections.

The greatest performance differences were caused by the challenge transmission methodology. Instead of using a QR code, sending only a random number as a challenge dramatically increased the performance of the execution time and the network usage. Sound waves, infrared technology, or motion detection can also be used as alternatives to a QR code in order to improve the performance of the authentication process.

In addition, the different device hardware specifications did not significantly effect the network usage. However, the execution time performance slightly increased while using higher performance devices during the authentication process. Moreover, the variety of the devices showed that the proposed authentication protocol can be implemented by using even slower devices.

Finally, none of the configurations significantly effected the battery life of the devices during the authentication process.

CHAPTER 6

DISCUSSION AND FUTURE WORK

In this chapter, several extensions will be addressed and future opportunities will be discussed.

Although many authentication schemes are being introduced, most of them are vulnerable to theft-based attacks. The new authentication protocol using two associated devices has been proposed to minimize attack surface over theft-based attacks. Several real-life examples of the authentication protocol were designed and implemented by using multiple sets of devices. Then, the authentication protocol was modeled with a high level protocol specification language and verified by a model checker in order to indicate the soundness of its secrecy and authenticity properties. Finally, the performance overhead of the protocol was evaluated to demonstrate the performance differences between multiple implementations.

6.1 Add/Remove Devices

A particular registration method was implemented and used in all configurations to pair devices with each other and the user. This registration phase can be done only once. To increase usability, it would be a very useful feature to allow users to add or remove additional devices into the system after the first device pairing process.

6.2 Continuous Authentication

Continuous authentication is an emerging topic in the security field. The main concern in standard user authentication schemes (e.g., username/password) is that after a certain

amount of time the user can become inactive. However, asking for authentication within a period of time could be used to track activity of the user.

In the proposed authentication protocol, if the system successfully authenticates the user, then that authentication remains active until the defined session time. However, an attacker can steal the authenticated device before the authentication session expires and get access to the system. To solve this problem, a continuous authentication approach could be integrated into the proposed implementations. The system could send a challenge to the first device every few minutes, and the user could perform this challenge with the second device in order to keep authenticated.

6.3 Future Experiments

Various experiments have been performed on the proposed implementation to demonstrate the performance overhead of the authentication system. During the experiments, the user interaction time was not considered. Using human participants and measuring their challenge performing times could be a good way to evaluate real authentication time with user interaction. Furthermore, similar experiments could be performed on other authentication schemes (e.g., username and password, physical token, two-factor authentication) to illustrate the performance advantages and disadvantages over different schemes.

6.4 Future Implementations

In Chapter 4, three particular implementations were presented. These authentication protocol implementations introduce solutions to distinct authentication problems. However, it would be useful to demonstrate more implementations to solve real life authentication problems.

Table 6.1 shows further possible implementations to solve common authentication problems in daily life. One general problem with cars is if the car key is stolen, an attacker also can steal the car itself. However, adopting the proposed authentication protocol could solve

Table 6.1. Further implementation ideas

Authenticator System	**First device**	**Second device**	**Challenge transmission method**
Car	Smartwatch	Smartphone	NFC
Home door	Smartphone	Smartwatch	QR code
Hotel door	Smartring	Smartwatch	NFC
Garage door	Car	Smartphone	Bluetooth
GPS navigator	Car	Smartphone	Bluetooth
Military facility	Smart necklace	Smartwatch	Bluetooth
Server	Laptop	Smartwatch	Image

the problem without extra overhead cost. For example, the car can represent the authenticator system. When the user gets close to the car, the car sends a challenge to the smartwatch, then the user taps the smartphone to perform the challenge. Finally, the smartphone sends the challenge back to the car and the car unlocks the doors and/or starts the car.

Another common problem is garage door or facility gate access. In this particular problem, the garage door represents the authenticator system and the user's car and smartphone act as the first and second devices. When the car gets close to the garage door, the garage door sends a challenge to the car. Then, the smartphone reads this challenge via bluetooth, and sends it back to the garage door. Finally, the garage authenticates the car and opens the door.

Alternatively, this authentication protocol can be implemented with any other lock system. The lock can act as the authenticator system and the user's multiple devices (e.g., a smart ring and a smart military necklace, a smart "dog tag") can act as the first and second devices. When the user comes within a threshold proximity of the lock, the smart ring will initiate the authentication protocol with the lock. Then, the lock sends a challenge to the smart ring. The smart ring sends the challenge to the smart dog tag. In this particular implementation, the protocol executes automatically, without user involvement, based on proximity to the lock.

Furthermore, more than two devices could be implemented during the authentication process. Two devices might not be enough for theft protection, especially when authenticat-

ing users to more sensitive information, such as, military source, intelligence agency source, etc. In this particular solution, a military base could house an authenticator system, and a user could use a smartphone, a smartwatch and a smart ring all together in order to complete authentication.

LIST OF REFERENCES

[1] Matt Bishop. *Computer Security: Art and Science*, volume 200. Addison-Wesley, 2012.

[2] Mark Stamp. *Information Security: Principles and Practice*. John Wiley & Sons, 2011.

[3] Anil K Jain, Patrick Flynn, and Arun A Ross. *Handbook of Biometrics*. Springer Science & Business Media, 2007.

[4] Lawrence O'Gorman. Comparing Passwords, Tokens, and Biometrics for User Authentication. *Proceedings of the IEEE*, 91(12):2021–2040, 2003.

[5] Jean-Baptiste Subils. Authentication Via Multiple Associated Devices. Master's thesis, University of South Florida, 2015. To appear.

[6] Cagri Cetin, Jay Ligatti, Dmitry Goldgof, and Jean-Baptiste Subils. Systems and Methods for Authentication Using Multiple Devices, April 22 2015. US Patent App. 14A079PRC.

[7] Messaoud Benantar. *Access Control Systems: Security, Identity Management and Trust Models*. Springer Science & Business Media, 2006.

[8] Steven Furnell. An Assessment of Website Password Practices. *Computers & Security*, 26(7):445–451, 2007.

[9] Kim-Phuong L Vu, Robert W Proctor, Abhilasha Bhargav-Spantzel, Bik-Lam Belin Tai, Joshua Cook, and E Eugene Schultz. Improving Password Security and Memorability to Protect Personal and Organizational Information. *International Journal of Human-Computer Studies*, 65(8):744–757, 2007.

[10] Robert Morris and Ken Thompson. Password Security: A Case History. *Communications of the ACM*, 22(11):594–597, 1979.

[11] David L Jobusch and Arthur E Oldehoeft. A Survey of Password Mechanisms: Weaknesses and Potential Improvements. part 1. *Computers & Security*, 8(7):587–604, 1989.

[12] David L Jobusch and Arthur E Oldehoeft. A Survey of Password Mechanisms: Weaknesses and Potential Improvements. part 2. *Computers & Security*, 8(8):675–689, 1989.

[13] Passwords Revealed by Sweet Deal, April 2004. http://news.bbc.co.uk/2/hi/technology/3639679.stm.

[14] Qinghan Xiao. Security Issues in Biometric Authentication. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 8–13. IEEE, 2005.

[15] Umut Uludag and Anil K Jain. Attacks on Biometric Systems: a Case Study in Fingerprints. In *Electronic Imaging 2004*, pages 622–633. International Society for Optics and Photonics, 2004.

[16] Anil K Jain, Lin Hong, Sharath Pankanti, and Ruud Bolle. An Identity-Authentication System Using Fingerprints. *Proceedings of the IEEE*, 85(9):1365–1388, 1997.

[17] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical One-way Functions. *Science*, 297(5589):2026–2030, 2002.

[18] EMC Corporation. RSA SecurID Hardware Tokens | Two-Factor Authentication, 2015. http://www.emc.com/security/rsa-securid/rsa-securid-hardware-tokens.htm.

[19] Blizzard Entertainment. Battle.net Authenticator - Battle.net Support, 2015. https://us.battle.net/support/en/article/battlenet-authenticator.

[20] Yubico. YubiKey Standard & Nano, 2015. https://www.yubico.com/products/yubikey-hardware/yubikey-2/.

[21] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology—CRYPTO'86*, pages 186–194. Springer, 1987.

[22] Manik Lal Das, Ashutosh Saxena, and Ved P Gulati. A Dynamic ID-based Remote User Authentication Scheme. *Consumer Electronics, IEEE Transactions on*, 50(2):629–631, 2004.

[23] Hung-Yu Chien, Jinn-Ke Jan, and Yuh-Min Tseng. An Efficient and Practical Solution to Remote Authentication: Smart Card. *Computers & Security*, 21(4):372–375, 2002.

[24] Dwayne Mercredi, Joseph Robinson, and Joachim Vance. Token Authentication System, October 17 2005. US Patent App. 11/252,040.

[25] Yen Sung-Ming and Liao Kuo-Hong. Shared Authentication Token Secure Against Replay and Weak Key Attacks. *Information Processing Letters*, 62(2):77–80, 1997.

[26] Geoff Brown. The Use of Hardware Tokens for Identity Management. *Information Security Technical Report*, 9(1):22–25, 2004.

[27] Guomin Yang, Duncan S Wong, Huaxiong Wang, and Xiaotie Deng. Two-factor Mutual Authentication Based on Smart Cards and Passwords. *Journal of Computer and System Sciences*, 74(7):1160–1172, 2008.

[28] Fadi Aloul, Syed Zahidi, and Wassim El-Hajj. Two Factor Authentication Using Mobile Phones. In *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*, pages 641–644. IEEE, 2009.

[29] Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. CAPTCHA: Using Hard AI Problems for Security. In *Advances in Cryptology—EUROCRYPT 2003*, pages 294–311. Springer, 2003.

[30] Jeff Yan. Bot, Cyborg and Automated Turing Test. In *Security Protocols*, pages 190–197. Springer, 2009.

[31] Philippe Golle. Machine Learning Attacks Against the Asirra CAPTCHA. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 535–542. ACM, 2008.

[32] Elie Bursztein, Matthieu Martin, and John Mitchell. Text-based CAPTCHA Strengths and Weaknesses. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 125–138. ACM, 2011.

[33] Carlisle Adams and Steve Lloyd. *Understanding Public-key Infrastructure: Concepts, Standards, and Deployment Considerations*. Sams Publishing, 1999.

[34] Mark Masse. *REST API Design Rulebook*. " O'Reilly Media, Inc.", 2011.

[35] Pivotal Software Inc. Overview of Spring Framework, 2012. http://projects.spring.io/spring-framework/.

[36] Hibernate Object/Relational Mapping, 2014. http://hibernate.org/orm/what-is-an-orm/.

[37] Alan Paller Dennis Kirby Bob Martin, Mason Brown. Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'), 2011. http://cwe.mitre.org/top25/CWE-89.

[38] The Apache Software Foundation. Apache Maven Project, 2015. https://maven.apache.org/.

[39] The Apache Software Foundation. Apache Tomcat, 2015. http://tomcat.apache.org/.

[40] Rich Helton and Johennie Helton. *Java Security Solutions*. John Wiley & Sons, Inc., 2002.

[41] D. Eastlake, 3rd and P. Jones. US Secure Hash Algorithm 1 (SHA1), 2001.

[42] Oracle. Java Security (Java Platform SE 7) | Oracle Documentation, 2011. http://docs.oracle.com/javase/7/docs/api/java/security/package-summary.html.

[43] Google Zxing | GitHub, 2014. https://github.com/zxing/zxing/.

[44] Gerald Madlmayr, Josef Langer, Christian Kantner, and Josef Scharinger. NFC Devices: Security and Privacy. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 642–647. IEEE, 2008.

[45] Google. Near Field Communication | Android Developers. https://developer.android.com/guide/topics/connectivity/nfc/index.html.

[46] The AVISPA Project, funded by the European Union in the Future and Emerging Technologies (FET Open) programme, Project Number: IST-2001-39252., 2003. http://www.avispa-project.org/.

[47] Luca Viganò. Automated Security Protocol Analysis with the AVISPA Tool. *Electronic Notes in Theoretical Computer Science 155 (2006) 61–86*, pages 64–66, 2006.

[48] David Basin Carlos Caleiro, Luca Viganò. Deconstructing Alice and Bob. *Electronic Notes in Theoretical Computer Science 135 (2005) 3–22*, pages 19–20, 2005.

[49] Google. Using DDMS | Android Developers, 2012. http://developer.android.com/tools/debugging/ddms.html.

[50] Patrick Dubroy. Memory Management for Android Apps. Google I/O Development Conference, https://www. youtube.com/watch?v=$_{C}ruQY55HOk$, 2011.