Using the Smart Card Web Server to Enhance the Security of Web Applications and the Web of Things

Lazaros Kyrillidis

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
School of Mathematics and Information Security
Royal Holloway, University of London
2018

Dedication

To my parents Dimitris and Zoe, to my late biological mother Asimina, to my brother Ioannis-Angelos, to my sisters Sofia and Despoina, to my brother-in-law Dimitris and to my niece Athanasia-Christina.

To my best friend Eugenia for the continuous support and patience.

Έχεις δει το τριαντάφυλλο
στη σκόνη του ατσαλιού
(ή πούπουλα κύκνων ποτέ·)
τόσο ελαφριά είναι η παρόρμηση
τόσο γαλήνια τα σκοτεινά
πέταλα του σιδήρου
εμείς που διαβήκαμε τη Λήθη

Έζρα Πάουντ, Τα Κάντος της Πίζας

Hast 'ou seen the rose
in the steel dust
(or swansdown ever?)
so light is the urging,
so ordered the dark
petals of iron
we who have passed over Lethe.

Ezra Pound, Pisan Cantos

Declaration

I, Lazaros Kyrillidis, hereby declare that this thesis and the work presented in it is entirely my own. Where I have collaborated or consulted the work of others, this is always clearly stated.

Lazaros Kyrillidis

Signed: ____________________

Date: ______________________

Abstract

The invention of the World Wide Web (Web) has changed forever the way society operates. Communication, shopping, entertainment, have all transformed and nowadays a large part of humanity depends on the Web for a number of everyday tasks.

In recent years, a new technology appeared. It is called the Internet of Things (IoT) and assumes the participation in a worldwide network of any device, despite its size, functionality or purpose. Both the Web and the IoT face a number of security concerns. The security of the Web is undermined by threats that aim at disrupting its normal operation, while for the IoT it is the use of proprietary protocols that weaken the interoperability between individual devices and the ease of tampering reduces its usefulness. Fortunately, the industry has knowledge and experience in developing a strong, tamper resistant device, the smart card. Smart cards are ubiquitous and strengthen a number of functions (payments, mobile telephony) and are the most secure token in mass production. Moreover, in recent years their interconnection capabilities have been further enhanced and this enhancement made available the hosting of the Smart Card Web Server (SCWS). The SCWS is a small web server running inside the smart/SIM card.

In this thesis we describe research that by using smart cards and the SCWS, over standardised protocols, we can enjoy tamper resistant solutions and enhanced security for the Web. In the same context, we consider a subcategory of the IoT, the Web of Things (WoT), that is using web-enabled protocols for the IoT and enhance its security by means of the combination of smart cards with the SCWS. The feasibility of the solution is depicted through four use cases, for which architecture and protocols are described and the necessary security analysis is conducted. Future work can demonstrate in practice that the SCWS with smart cards can enhance the security of the Web and the WoT.

Acknowledgement

Professor Keith Mayes guided me throughout this adventure with patience and perseverance. Even when I lost faith, he still insisted, believed in me and this thesis would not have been completed without him. I cannot thank him enough.

A big thank you to my brother, Ioannis-Angelos, that kept me going in the most crucial moment.

Finally, I need to thank everyone who believed in me and more importantly those who did not.

Publications

Distributed E-Voting Using the Smart Card Web Server, 7th International Conference on Risks and Security of Internet and Systems (CRiSIS 2012), Cork, Ireland

Remote E-Voting Using the Smart Card Web Server, CRiSIS 2012 Special Edition of International Journal of Secure Software Engineering (IJSSE), Vol.5(1), IGI Global, 2014

Card-Present Transactions on the Internet Using the Smart Card Web Server, 12th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2013), Melbourne, Australia

Virtual World Authentication Using the Smart Card Web Server, International Symposium in Security in Computing and Communications (SSCC'13), Mysore, India

A Smart Card Web Server in the Web of Things, SAI Intelligent Systems Conference 2016 September 21-22, 2016, London, UK.

Contents

List of Figures

List of Tables

List of Abbreviations

3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
BIP	Bearer Independent Protocol
(D)DoS	(Distributed) Denial of Service
ETSI	European Telecommunications Standards Institute
FAP	Full Administration Protocol
FIPS	Federal Information Processing Standards
GSM	Global System for Mobile Communications
HTML	Hypertext Markup Language
HTTP(S)	Hypertext Transfer Protocol (over SSL)
ICCID	Integrated Circuit Card Identifier
IoT	Internet of Things
IP	Internet Protocol
ISO/IEC	International Organization for Standardization/International Electrotechnical Commission
MNO	Mobile Network Operator
NFC	Near Field Communication
OMA	Open Mobile Alliance
OTA	Over-The-Air
OTP	One-Time Password
OWASP	Open Web Application Security Project
PAN	Primary Account Number
PCI DSS	Payment Card Industy Data Security Standard
PIN	Personal Identification Number
RAS	Remote Administration Protocol
RSA	Rivest–Shamir–Adleman Algorithm
SCWS	Smart Card Web Server
SHA	Secure Hash Algorithm
SIP	Session Initiation Protocol
SMS	Short Message Service
(U)SIM	(Universal) Subscriber Identity Module
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UICC	Universal Integrated Circuit Card
USB	Universal Serial Bus
WoT	Web of Things
WWW	World Wide Web

Chapter 1

Introduction

1.1 The World Wide Web

The World Wide Web (Web) has been a reality for almost three decades. The way that we communicate, are entertained, shop and are informed has changed since the invention of the Web by Tim Berners Lee in 1989 [1]. Berners Lee may have never predicted the success of his invention, but the Web is an important part of modern societies. The invasiveness of the Web is obvious in most traditional human actions. Nowadays there are banks that operate entirely on the Web, gigantic shops with literally no physical presence (thus no 'Bricks and Mortar') like Amazon or eBay, banking on the Web is becoming a daily task for a lot of people, even the way people flirt and date has changed drastically.

Part of the Web's success, if not the most important, is due to the fact that it is based on well-established standards, especially when it comes to the protocols that facilitate communication between entities on the Web. This allows every modern device - laptops, PCs, smartphones, tablets, watches - to have the ability to connect to remote web applications and retrieve information that they then display back to the user. The user just needs a web browser, like Google Chrome, Mozilla Firefox or Opera to see and interact with this content. The way that this content is retrieved is based on the client-server model where the client (i.e. the user's browser) is requesting information from an entity (the web server) that hosts and/or processes this information or retrieves it from other entities (e.g. in the form of web services). In this instance the server itself is acting as a client receiving information from another server [2].

The Web (and the Internet in general) was created in an era when people that had access to it were few and the access was part of their research and work. This quickly changed.

The Web is now a field where malicious entities can also profit and this is indeed what is happening in reality. Security problems are numerous and they are getting worse every day. These issues vary in nature and complexity and affect the trust that certain parts of the society have in the Web.

This insecurity of the Web, the attacks that aim at disrupting the normal operation of processes running on it, are worsening by the fact that non-standardised protocols and insecure devices are used in the IoT (as we will see later in this thesis). Distributed Denial of Service attacks (DDoS) are getting more difficult and expensive to deal with, since they are now stretching the boundaries of our defenses [3]. Although the average DDoS attack peaks at approximately 1.5 Gbps, there have been instances where attacks have peaked at over 1Tbps. Mitigating these attacks is very difficult and even if it were possible, it would prove to be a very expensive task to undertake. Instances of criminals or activists trying to disrupt normal services are common [4].

Since mitigating these attacks is monetarily an expensive exercise for companies and organisations, in the past they have not invested in technologies or infrastructure to do so. Lately however, in light of recent attacks on large organisations, these entities are beginning to address security issues and are incorporating their reputations into the equation when investing in the necessary equipment and personnel.

For criminals to conduct these kinds of attacks they need a large number of devices under their control. These devices can then on a certain day be instructed to start a DDoS attack against the victim (in society, individuals or organisations) and have traditionally used PCs or laptops infected by malware, but now the criminals have discovered another target. This target is comprised of insecure devices and they are herding them in the thousands, infecting them with malicious code and forcing them to participate in DDoS attacks. The main reason that the members of the criminal community are able to get access to these devices is the weak authentication methods and the fact that the majority of end users do not change the default login credentials to access these devices. This thesis will discuss this problem later in detail.

How well authentication methods work on the Web is thus a topic of continuous debate. It is well proven and widely accepted that passwords are now a fairly insecure method of authentication [5] and discussions on how to stop using passwords as an important defence mechanism are constant. Risk based authentication [6], adaptive authentication [7], context aware authentication [8], continuous authentication [9], biometric authentication [10], multi-factor authentication [11] are some of the proposed answers. However, all these methods have their own challenges and difficulties and are not always straightforward to implement.

Another important problem on the Internet, is the security of transactions on e-commerce sites. There have been numerous attacks against merchants that have resulted in the data theft from millions of credit/debit card holders. Stolen cardholder information is sold on the Dark Web and depending on the type of card, fetching more than £30 per card [12]. An important factor, that is not directly affecting the end user, is the difficulty for merchants to deal with the security controls that the Payment Card Industry Data Security Standard (PCI DSS) places on merchants [13]. Depending on the amount of credit/debit card payments that the merchant receives, it may be required to go through the difficult process of complying to a large number of PCI DSS requirements (or all depending on the PCI DSS level that the merchant belongs to), a task that has been proven to be onerous and very expensive.

1.2 Internet of Things/Web of Things

In recent years, another technology made its appearance. It is the Internet of Things (IoT), which in reality is a network of connected devices of any kind [14]. These devices, called ‘Things’, act independently or as a group to provide specific services. The basic idea is that over time, everything will become connected to a network, Local (LAN) or Wide Area Network (WAN) which could be global. These interconnected devices will provide all sorts of services, e.g. a sensor to detect environmental changes or a smart bulb activated by an application, to smart fridges that will inform their owner about their contents or even a whole house infrastructure that will allow full control of the entire house remotely.

One would assume that the industry would have learned from all the problems that the traditional Web has faced (and is still facing) and would have applied the necessary controls on the newer technology that the IoT is. Hence, as mentioned earlier in this chapter, this did not happen. It is a wide problem that IoT devices have very weak authentication methods and that they use proprietary protocols for communication and although weak authentication methods lead to obvious problems, the use of proprietary protocols may lead to false perception of security since the things cannot inter-communicate with other things that do not speak the same protocols. Also using a proprietary protocol that has not gone through public scrutiny is a problem to security. In an interconnected world, it would have also been assumed that the things would have been easily accessible using a standard browser and that developing applications that would incorporate these things would have been quite straightforward. Unfortunately this is not the case. The result is that we now see isolated islands of things that are not easily accessible from anything that is not manufactured by the same company. Interoperability is an important factor for the success of the IoT. At the same time the security of this technology is still very weak.

To this extent, research has been conducted for devices to be able to communicate over well-established protocols instead of proprietary ones [15]. This reasonable idea is for devices to communicate over web protocols that have been under extensive scrutiny in terms of security and performance, but are also extremely well supported by every device that has basic communication abilities. This idea is called the Web of Things (WoT) and we can consider it a sub-category of the IoT that utilises web technologies and protocols for communication.

Unfortunately, although the WoT aims at solving (part) of the interconnection problems, the security problems that exist in the traditional IoT do not disappear in the WoT. Unauthenticated devices, with minimal levels of security still exist. The problem is intensified by the fact that these devices can be found in the hands of people that do not have the expertise in using them securely, thus making these devices more prone to attacks by malicious entities that try using them for their nefarious actions [16]. This thesis will propose a solution to this problem.

1.3 Smart Cards

However, the industry has the knowledge and experience of creating small electronic devices that are extremely tamper resistant. These devices support everyday functions and they are called smart cards/Subscriber Identity Module (SIM) cards [17]. Another important and very useful property of a smart card is that it can be securely personalised and in certain cases be managed post-issue, thus giving added value to the user.

A smart card is a hardware token that hosts an embedded microprocessor chip, Random Access Memory (RAM), Electrically Erasable Programmable Read-Only Memory (EEPROM), and Read-Only Memory (ROM) or more recently, Flash memory. A smart card can be used in many ways: as a credit/debit card for financial transactions, as an identity card to access restricted areas, as a ticketing solution or inside mobile phones to assist with cryptographic operations or to provide secure storage. The most significant security advantage that a smart card offers is that it is a tamper resistant token that defends against software and physical attacks.

A smart card can make a specific attack either too difficult and/or expensive or even infeasible for the perpetrator to launch in terms of equipment or expertise needed, or even impossible. In a worst case scenario, a smart card will not be able to resist a specific attack, but will provide proof of the attack. The above tasks that a smart card offers or assists with, are very important and widely used. However, as this thesis will propose, we anticipate that smart cards can be used in even more productive ways, especially when we consider modern smart cards that are equipped with advanced capabilities: these advanced capabilities come in the form of more powerful chips, higher memory capacities, all supplied in small form factors/sizes [18].

The integration of smart cards in the Internet and the IoT/WoT is the main idea behind this thesis and will be demonstrated through specific use cases.

1.4 Research Question

To help mitigate risks that arise from the above threats and insecurities, the idea to integrate a smart card into the wider Web, was developed. The industry acknowledges that a token as secure as a smart card can be used to secure web applications and at the same time provides a pleasant experience to the end user.

For this to occur the players in the industry designed and adopted a new type of smart card that is easier to integrate with the Web, is faster and development of applications with its use are much easier. These cards are the Java Card v3.0 Connected Edition [19]. They natively support web-based communication and protocols, they implement a TCP/IP stack, they support a larger version of the Java programming language compared to traditional Java Card (e.g. multithreading, multi-dimensional arrays), meaning that they are easier to integrate with the Web and it is much easier to develop applications for them in comparison to the traditional Java Card. These cards are also able to support and host a small entity called the Smart Card Web Server (SCWS). The SCWS is a stripped down HTTP web server that is able to serve web pages to the user and is easier to integrate in the wider Web. Applications running on the SCWS are running inside the secure environment that is offered by smart cards, achieving two goals: interoperability and security.

This thesis tries to provide an answer to the following question: how can we use a Java Card v3.0 Connected Edition with a Smart Card Web Server to enhance the security of the aforementioned web applications and to assist in a more secure authentication in the WoT? This will be accomplished by presenting a number of use cases and protocols that use the SCWS to mitigate specific threats and risks:

1. The first use case will use e-voting process as an example of how we can distribute the process among a number of ‘personal voting websites’ to help defeating (D)DoS attacks.

2. The second use case aims at showing how we can use our proposal to ease the burden of compliance with PCI DSS by demonstrating an alternative way for purchases that will make e-commerce transactions safer, without sacrificing user convenience.

3. The third use case aims at providing a solution to the authentication to web applications, by using a virtual world as an example. The authentication method presented in this use case can also be used in risk based authentication scenarios, whenever an advanced level of authentication is required.

4. The fourth use case will extend our research to the WoT environment, by trying to recommend an authentication solution that will utilise a distributed single sign-on infrastracture provided by the SCWS.

All four proposals will use a Java Card v3.0 Connected Edition, hosting the SCWS, utilising standardised protocols and the tamper resistance offered by modern smart cards. At the same time the interoperability offered by direct web access (based in the fact that these cards may also implement a TCP/IP stack), will turn smart cards into network nodes making it easier to integrate into the wider Web or the WoT. The use of standardised protocols offers confidence that our solutions will not use 'security through obscurity'.

The task was not without challenges. This thesis approaches the topic from a theoretical point of view due to lack of equipment conforming to the required standards. However, formal verification and analysis of the proposed protocols is conducted using a relevant tool, Scyther. We believe that future research can apply our theoretical protocol designs and ideas in a real, working Java Card v3.0 Connected Edition.

1.4.1 Basic Assumptions

This thesis will use the following assumptions:

1. Java Card v3.0 Connected Edition is referenced throughout the thesis. This Java Card version is more suitable for the purposes of this thesis and is preferred, than Java Card Classic Edition.

2. The SCWS, as defined in the specification, has to support HTTP/1.1.

3. The SCWS has to support the basic TLS versions: TLS 1.0, 1.1 and 1.2, although the recommendation of this thesis is to use the latest (and more secure) TLS 1.2 version.

1.5 Structure of the Thesis

After this introduction, this thesis is structured as follows. Chapter 2 discusses smart cards and Chapter 3 talks about web servers and the Smart Card Web Server. Chapter 4 deals with the protection of e-voting from DDoS attacks by distributing the voting process across multiple SCWS-enabled smart cards, Chapter 5 discusses the use of the

SCWS to secure e/m-payments, while Chapter 6 expands the use of the SCWS on a virtual world environment, dealing with authentication concerns. Chapter 7 pushes the SCWS into the IoT/WoT realm by introducing the concept of security modules, i.e. SCWS-enabled smart card chips that may be attached on things to accommodate authentication on an WoT environment. Chapter 8 concludes this thesis.

Chapter 2

Smart Cards

Contents

2.1 Introduction

A smart card is a device that hosts an embedded integrated circuit/chip. This chip provides smart card with the ability to perform calculations, store and execute code and participate in cryptographic operations. However, not everything that hosts a chip can be called a smart card - what differentiates smart cards is the fact that they are relatively small in size and they do not possess power, but they rather require an external entity to provide them with energy to operate. Their physical characteristics are defined in ISO7816-1 [20]. Another important factor that sets smart cards aside and puts them in a class of their own, is that they are tamper resistant, whereas very rarely other chips are, e.g. the ones that are used in common PCs/laptops or other devices do not provide tamper resistant capabilities.

This ability is proven to be an important factor for the success that smart cards enjoy. Additionally, smart cards have some form of operating system (OS), however, this operating system can be considered a relatively primitive one, compared to modern operating systems. This OS is able to perform basic RAM operations (access to memory, move data in and out of the RAM) and to control other parts of a smart card, like flash memory, ROM and EEPROM. This OS is also able to host certain types of applications and provide a secure execution environment for cryptographic operations, although the latter is often or can be taking place inside a cryptographic co-processor (thus not inside the 'main' smart card processor), a processor specially designed and dedicated for cryptographic operations. This cryptographic co-processor is usually accessible from the smart card OS and assists the code execution on the main chip, by enabling much faster execution of symmetric and asymmetric cryptographic functions.

2.2 History of Smart Cards

The first patent for a smart card was registered in 1974 by the French engineer Roland Moreno - Moreno is considered the inventor of smart cards [21], however the first patent for a chip card was filled a few years before that, in 1968 by two German engineers, Jürgen Dethloff and Helmut Gröttrup and also in 1970 the Japanese Dr. Kunitaka Arimura filled the first patent for the smart card concept. Moreno was the one that patented the integrated circuit memory card which is the idea employed by modern smart cards. The first patent that Moreno filled was for a ring embedded with a chip, but this was proven not successful, so he made certain modifications in 1975 by adding the chip on a plastic card. This idea was very innovative and gained popularity, first in France and later to the rest of the world and up to present day.

In 1978, Michel Ugon of Honeywell Bull, invented and patented the microprocessor smart card. This patent was bought by SGS Thomson and Schlumberger and along with Bull CP8, led the mainstream market of smart cards. Motorola followed shortly afterwards.

In 1983, France Télécom used Moreno's smart card as a basis for its Télécarte pay phone, as a means to pay for calls using a prepaid, memory smart card. 1984 was a very important year in the history of smart cards, with a trial conducted by the French Postal and Telecommunication services about the use of ATM bank cards with embedded chips. A few years later, in 1992, the French banking industry presented the Carte Bleue, a debit card for national use. This concept was later expanded to the rest of the world.

At the beginning of the 20th century, Schlumberger bought the CP8 patent and the whole department from Bull, merging it with its internal department creating Axalto. In 2006 Axalto merged with Gemplus (the other worldwide leading smart card manufacturer), creating Gemalto, which is one of the leaders in smart card manufacturing until present day.

2.3 Smart Card Sizes

Smart cards can come in various types and sizes [18]. The most common type that people use daily is a rectangular plastic ‘card’ that is approx. 85.60 by 53.98 by 0.76 millimeters. This is the size for cards used in banking operations or for identification purposes (access cards). A special category of smart card that is in common use today are the so called Universal Subscriber Identity Modules (USIM) cards that are mostly known as SIM cards (for their use please see later in this chapter). Full sized SIM cards (also called 1FF cards) have the same size as ‘normal’ smart cards, but these are nowadays almost extinct. Modern SIM cards come in three sizes:

1. Mini-SIM (2FF): 25.00 by 15.00 by 0.76 millimetres
2. Micro-SIM (3FF): 15.00 by 12.00 by 0.76 millimetres
3. Nano-SIM (4FF): 12.30 by 8.80 by 0.67 millimetres.

2.4 Basic Standards

Smart cards are well standardised entities, their attributes are well defined in various standards and this greatly promotes their use in a number of applications. This thesis will not be involved with low-level technical attributes, however for a better understanding of the thesis, a short description of the basic standards that are used throughout, will be provided.

2.4.1 ISO/IEC 7816

ISO/IEC 7816 is arguably the most important of all smart card standards [22]. It is divided into 14 parts that define the physical characteristics of the cards, their electrical interfaces and protocols used for communication. It also defines commands that are used to communicate with the smart card and the logical structure of the cards i.e. the file system and the types of files.
ISO 7816 applies to contact and contactless cards.

2.4.2 ISO/IEC 14443

ISO/IEC 14443 defines the necessary interfaces for the communication with a ‘close proximity’ contactless smart card [23]. Cards compliant to ISO/IEC 14443 operate at 13.56 MHz and have an operational range of up to 10 centimeters (3.94 inches). Near Field Communication (NFC) cards are based on ISO/IEC 14443 and in general this standard is the basis for a large number of applications used in the financial or transport field.

2.4.3 ETSI TS 102 223

European Telecommunications Standards Institute (ETSI) Technical Specification (TS) 102 223 'Smart Cards; Card Application Toolkit (CAT)', defines an interface between the Universal Integrated Circuit Card (UICC) and the terminal [24]. The Card Application Toolkit is a set of commands that the UICC can use - the UICC must support at least one application for access to the network. For example, this application can be the SIM for 3G or 2G networks or the (U)SIM for 3G networks. This application is called Network Access Application (NAA).

2.4.4 ETSI TS 102 483

ETSI TS 102 483 [25] defines the way that a UICC can establish an Internet Protocol (IP) connection with a terminal. The connection will be able to route IP packets and the specification establishes and describes the way that resources inside the UICC can be accessed from the terminal using this way.

2.4.5 ETSI TS 102 600

ETSI TS 102 600 describes a Universal Serial Bus (USB) interface between a terminal and a UICC [26]. The USB interface is very important in instances of an application that requires a fast data communication protocol (as we can see later in this thesis). The specification aims at providing interoperability for the communication between the terminal and the UICC.

2.5 Types of Smart Cards

We can separate smart cards broadly in three categories: contact, contactless and hybrid cards. The differentiating factor that separates and categorises cards in these three categories is based on the way that they communicate with external entities, i.e. with smart card readers. Early smart cards communicated only through being in contact with a smart card reader, i.e. the card had to be inserted inside the reader. The reader provided power to the card and the ability to communicate with other systems, e.g. a laptop or a bank end bank system. Although this type of function is still widely used for smart cards, in recent years smart cards acquired the ability to communicate with readers without physical contact to them, i.e. contactless. The card has to either be in very close proximity with the reader (almost touching it) or it can operate from a distance, usually of a few centimetres. The latter is usually an NFC or Bluetooth enabled card. Hybrid or dual interface cards can function either as contact or contactless cards - the operating mode depends on the reader and its abilities (thus if the reader can support contactless connections).

Another way to differentiate smart cards is by categorising them based on 'how smart' they really are. Thus we can speak of memory cards and cards with a Central Processing Unit (CPU). The first category is comprised of cards that contain a memory where a value is stored. This value may or may not be changeable (thus the memory is writable or not) and is more frequently used as phone or prepaid/rechargeable cards. Phone cards for example can be bought with a predefined amount of money and can be used until this amount reaches zero. At this point the card is useless and cannot be used again. A rechargeable card can be found e.g. in hotels where the guest tops up the card with a certain amount of money for purchases on premises and can recharge the card if it runs out of money.

Cards with a CPU on the other hand are generally, 'smarter'. The fact that they possess a CPU provides them with the ability to run and execute complex calculations.

A smart card is able to even run cryptographic operations (symmetric or asymmetric), but sometimes for this type of functionality a secondary processor (crypto co-processor) is used (a fact that raises the price of the smart card). The crypto co-processor is a specially designed processor that can natively run crypto commands - these commands are usually executed in native code that runs directly on the processor, instead of being executed as a program/function on top of an operating system. Usually, this type of smart card possesses an operating system. This OS is different than 'normal' OSes (found on other devices, like laptops or PCs), in that it does not possess a user interface and generally the user cannot access and execute commands on it. However, this OS is capable of executing tasks like 'normal' operating systems do, for example it controls access to RAM and provides an environment for the execution of on-card applications.

2.6 Smart Card Operating Systems

There are two dominant smart card operating systems:

2.6.1 Java Card

A Java Card as the name implies, is a smart card that is able to run applications written in the Java programming language. In reality, these applications are utilising a subset of Java, since supporting the full version of Java is impossible, given the resource constrained nature of the cards.

The idea behind the use of Java Card is identical to the one that led to the creation of Java, as a programming language, itself. The ability of a Java program to be run on every smart card platform that supports the execution of Java programs without the need for re-compiling it. In practice this is achieved in the same way that a traditional Java program is executed. The underlying Java Card Virtual Machine (JCVM) runs below the Java Card API (JCAPI). The JCAPI is a set of libraries and functions that a Java Card application, which in the realm of Java Cards is called an applet, can call during its execution and can be seen as a layer running on top of the JCVM, exposing Java Card functionality to the on-card applications. The purpose of the Java Card Virtual Machine is to provide the environment that will enable Java Card applets to run, without having to worry about portability issues, thus implementing the 'Write Once, Run Anywhere' motto of the Java language. The JCVM is the mechanism that loads Java classes and executes them.

Both the JCVM and JCAPI comprise the Java Card Runtime Environment (JCRE). The JCRE defines the requirements under which a Java Card application can run.

The development of a Java Card applet is not radically different compared to the development of 'normal'/conventional Java applications. A programmer writes the Java Card code that then compiles. This produces a number of Java Card classes. Usually the applet is not executed on an actual Java Card, but rather the programmer is using a simulation tool running on a PC. The simulation tool provides an estimate of the execution speed and will provide indications as to whether the applet will be able to run on a real card. Running on a simulation, allows for easier debugging and correction of programming errors.

If the simulation satisfies the requirements and expectations of the programmer, it is converted to a CAP (converted applet). For this purpose, the programmer will use a Java Card Converter, which is a tool specifically designed to combine all classes into a Java Card package. This stage also includes the verification of the CAP file by using a bytecode verifier that checks the CAP file and ensures that it has the correct format. This process additionally identifies and stops any malicious application that wants to find its way to the smart card.

Once this conversion is finished, the CAP is ready to be transferred to the card. This is the 'download' phase. Usually, a card reader attached to a PC is used for this purpose and a special installation tool must be running on the PC for this task to be completed. The installation tool 'downloads' the CAP onto the card and an on-card installation application prepares the applet for execution on the Java Card. The virtual machine is now ready to run the applet.

Each of the applets is run inside a 'Security Context'. Indeed a firewall separates applets and isolates them into their own memory sections, blocking any access to this context from other applets or themselves to access contexts where other applets reside. Applets that were developed by the same entity, can also access data stored in other contexts, as long as they share the same cryptographic key material.

2.6.2 Java Card v3.0

Java Card v3.0 is available in two different versions:

- Java Card v3.0 Classic Edition, which is an evolution of the Java Card v2.2.2 version and it is destined for smart cards of lower specifications [27].
- Java Card v3.0 Connected Edition, which is an advancement in smart card technology. Although it is compatible with applications/applets written for older smart cards or the Classic Edition, it introduces a number of improvements [19]. This version is destined for smart cards of advanced capabilities, with bigger memory and higher processing power.

2.6.2.1 Java Card v3.0 Connected Edition

Java Card v3.0 Connected Edition is a network-oriented version of smart card technology, that allows for direct integration with TCP/IP networks and the execution of web applications. Applications running on Java Card v3.0 Connected Edition can directly integrate with the Web and other IP networks and be accessed over standardised protocols like HTTP/HTTPS.

A Java Card v3.0 Connected Edition supports a wider version of the Java technology:

- It allows for multiple threads to run concurrently (multithreading).
- It supports a wider version of the Java programming language, e.g. multi-dimensional arrays, strings, integers, etc.
- It uses a garbage collector to remove session data that are no longer needed.
- It allows for on-card bytecode verification, which allows for the application's class files to be verified on-card, whereas with most of other versions of Java Card, the verification happens off-card.

Java Card v3.0 Connected Edition additionally offers another important advancement. As mentioned above, it is a smart card fully capable to integrate directly with TCP/IP networks and offer services over web applications. This is done through the implementation of a TCP/IP stack on the card and the hosting and execution of web applications that are running directly on the card. For the latter, the Smart Card Web Server is used (please see Chapter 3 for more information).

The ability for a smart card to directly integrate in a network, allows for exciting opportunities as the card effectively becomes a network node directly accessible by other network nodes.
Java Card v3.0 Connected Edition is the Java Card version that is assumed throughout this thesis.

2.6.3 MULTOS

The second most important smart card OS is MULTOS [28]. The development of this OS is overseen by a consortium of companies that includes card manufacturers, application providers, payment card schemes, etc. Everyday tasks relating to MULTOS are run by MAOSCO - MAOSCO is also responsible for the development of OS specifications, publication of white papers and general representation of the OS to the industry. As with Java Card, MULTOS is using a virtual machine to run applications in the MULTOS Executable Language (MEL). Similarly to Java Card, applications are developed and tested off card by using various development tools. For this, developers can write their code in a programming language and then this code is translated/compiled into the MULTOS specific language, MEL. The derived MEL bytecodes are then transferred onto the card and are run by the MULTOS virtual machine. This way and for as long as there exist compilers that can translate code developed in a programming language to MEL, programmers can write their code using their preferred language. Their programs can then be run on any MULTOS card without a problem. However, in practice two main languages are used (C and Java).

The application is signed by the application provider (using their private key). The public key is transmitted to the card issuer and through them to MULTOS and more specifically to the MULTOS Key Management Authority (KMA) that plays a central part in the whole scheme. MULTOS signs the public key of the application provider and creates a certificate that is loaded on the card in the form of the Application Load Certificate (ALC). Another certificate that is created and also stored on the card is the Application Delete Certificate (ADC) that is used when the application needs to be deleted from the card. MULTOS public keys are then inserted into the Application Load Unit (ALU) along with the code and other data provided by the application provider. The Application Load Unit is a package that the application provider uses to pack the application's data and code with the aforementioned keys that are then loaded on the card. An application is allowed to run on the card, if the verification process succeeds. The responsible entity for providing all this cryptographic assistance, is the MULTOS KMA.

The above process ensures that applications can be loaded even in insecure environments, because the protection and integrity of the code is ensured by this architecture.

As is the case with Java Cards, each application is separated by the rest using a firewall. Applications cannot access memory, information and data that belong to other applications and this ensures the integrity and confidentiality of the execution environment. Because of this architecture, MULTOS is considered a secure smart card OS and is generally preferred in cases where a higher level of security is desirable.

2.7 Main Components of the Smart Card

A smart card is comprised of the following parts [18]:

2.7.1 Central Processing Unit (CPU)

A CPU is the central point of intelligence for a smart card. As with a traditional Personal Computer (PC) a CPU is the part that controls the execution of commands. Traditionally, a CPU was an 8-bit microcontroller, but in more recent years 16-bit and even 32-bit are more common. The increase in processing power allows a smart card to execute more complicated commands and thus, to host more advanced applications (or applets in the case of Java Cards).

When a smart card is required to participate in cryptographic operations, it usually bears a second CPU. This is a special purpose CPU that is specially customised to run cryptographic operations. The customisation is due to the fact that all cryptographic operations do not execute on the application or operating system level, but rather are instructions written in native language and execute directly on the hardware. This means that these operations can run much faster. This CPU is usually called crypto co-processor. Whenever a complex cryptographic operation needs to run, the main CPU off-loads the entire execution to the crypto co-processor that executes the cryptographic operation(s) and returns the result to the main CPU for further processing.

2.7.2 Memory

A smart card has the following types of memory. These memory types are:

2.7.2.1 Read Only Memory - ROM

This is the type of memory that hosts the operating system and other code that is not changed for the lifetime of the card. Its contents cannot be changed, as it is a non-volatile type of memory, which means that its contents are retained even when the card does not receive power.

2.7.2.2 Electrically Erasable Programmable ROM - EEPROM

This is the type of memory that hosts the code for the applications that will execute on the card. It also provides the storage for content that needs to be stored on the card, e.g. phone numbers and other information. As such it is a non-volatile type of memory and it comes in sizes from 128 bytes to 1MB.

2.7.2.3 Flash Memory

A faster alternative to the above is a newer type of memory, flash memory. Flash memory can be used as a non-volatile memory space, replacing slower alternatives (e.g. EEPROM) offering faster read access times.

2.7.2.4 Random Access Memory - RAM

This is the smallest type of memory that exists on a smart card and it is used by applications running on the card as the place where fast computation can occur. Due to the fact that it is quite fast, it is able to execute commands with the minimum delay. It is a volatile type of memory and loses all its contents when the card is not powered up.

2.7.3 File Structure

According to ISO 7816-4 [29], a smart card has a hierarchical file structure. The three types of files that exist on a smart card are the Master File (MF), the Dedicated File (DF) and the Elementary File (EF). At the top of the file system sits the MF which, unarguably, is considered the root under which the rest of the smart card file system exists. A MF is selected after the initialisation of the smart card and contains every other file present on the smart card. It cannot be deleted.

On a second level, there are a number of 'intermediate' files, called Dedicated Files (DF). A MF is a special type of a DF. A DF may contain other DFs and theoretically there is no limit on the number of DFs that can exist under a specific DF, but in practice due to memory limitations there are only 2 levels of DF files.

The last level of files are the Elementary Files (EF). They are used to store data and depending on the application they have a specific file structure. An EF is the most basic type of file and cannot contain other files. There are 'external' or working EFs that are used for application storage and are accessible by external, to the card, entities (e.g. applications running on the phone) and internal EFs that contain data used by the operating system and are accessed by it only.

2.7.3.1 Data Structure

EF files have four different types for data structure:

1. Linear fixed: data are grouped into records of a predefined size.
2. Linear variable: data are grouped into records of a variable size.
3. Cyclic: it can be considered a special type of Linear Fixed type as it is of fixed size. However, when a cyclic file is full the first record is overwritten. This type of structure was used to store Short Message Service (SMS) messages, where whenever the file was full, the first SMS was replaced by the last one.
4. Transparent: it is a block of data without a data structure. Data are accessed as a reference/offset from the beginning of the data block.

2.8 Smart Cards Main Uses

Smart cards can be used in a number of different ways. Depending on the use case, smart cards of a certain type can be used (e.g. memory cards, or cards with a chip) [17].

2.8.1 Payphones

Payphone cards were very popular especially during the 90s. Users were buying prepaid cards that they could then use with payphones. Once the credit on the card ran out, the user had to buy another card.

2.8.2 Mobile Communications

A special category of smart cards can be found inside a phone and they are the called Subscriber Identity Module (SIM) cards. These cards provide cryptographic assistance to mobile communications, since they can securely store cryptographic keys that are used to encrypt the communication between the phone and the cell tower and also help identify the subscriber to the network. The SIM cards can also be used as application hosts through the existence of the SIM toolkit (where applications are running inside the SIM and could be accessed from the phone) or secure storage e.g. for contact lists. A large large number of smart cards in circulation, is comprised of SIM cards [30]. For more information on SIM cards, see the relevant section later in this chapter.

2.8.3 Banking

Another very important usage for smart cards is when they are used for banking purposes. Most of the debit or credit cards in a number of countries are now smart cards, whereas the old, swipe cards are now being replaced. The tamper resistance abilities of smart cards provide an excellent choice as alternatives to cash. Debit or credit cards can provide the environment for cryptographic operations to securely execute and their ability to provide exactly this secure operating environment gives them an important place for banking operations. They can authenticate themselves to the reader, but can also authenticate the reader as being genuine, thus avoid communicating with malicious readers.

2.8.4 Prepaid Cards for Retail

A smart card can also be used to store a prepaid amount of money that can be used towards purchases, e.g. visitors of a hotel can be provided with a prepaid smart card that they can then use against purchases inside the hotel. These cards can be ‘recharged’ with money, thus providing the person with an easy way for purchasing products. A special case for this type of card is transport cards (e.g. Oyster in London). These cards can be recharged with a specific amount of money and when the value is zero or below a threshold, they can be topped up automatically or using special machines. The user can touch her card on the station gates and depending on the monetary value that already exists on the card, determines whether the automatic gates open or not

2.8.5 ID Verification and Access Control

As mentioned before, smart cards possess the necessary computational power to run cryptographic operations and to authenticate themselves whenever needed. This ability can be exploited when smart cards are used for identification purposes. The card stores authentication details for the user in question and can provide these to legitimate readers (since it can participate in mutual authentication operations, meaning it can avoid malicious readers). The card can store from a single password to private asymmetric keys and even biometric information that can then be used to reliably identify the user.

2.9 Smart Cards Certification

Smart card manufacturers ensure the security of their products by certifying them against specific frameworks. The most prolific are Common Criteria (CC) and Federal Information Processing Standards (FIPS).

2.9.1 Common Criteria

This certification framework is a globally recognised method of evaluating the security of smart card implementations [31]. An evaluation based on CC will prove the security capabilities of smart cards, physically and logically. A CC evaluation can be used to compare different products and provide assurance to customers that a certain product meets specific security requirements.

A protection profile is a group of security requirements for a specific type of smart card and is the basis for CC certification. A protection profile defines the Target of Evaluation (ToE), which basically is the subject of evaluation. A protection profile will then provide the requirements that the specific evaluation will be based on e.g. certifying a firewall or an antivirus requires different requirements thus different protection profiles.

The evaluation assurance level (EAL) provides a measurement of how well the specific product fulfils the security requirements as defined in the protection profile. The product then is assigned a value that categorises it according to the CC levels (EAL 1-7). Most smart cards are evaluated to EAL4+.

A security target (ST) is a specific description of the ToE, the threats that it faces, implementation of specific countermeasures against these threats and the level of certification that the product achieved.

It must be noted that CC do not just apply to smart cards only, but also to software and hardware implementations can be certified against this framework.

2.9.2 Federal Information Processing Standards (FIPS)

FIPS is a standard developed and maintained by the US government and has a different role compared to Common Criteria. FIPS has four levels of certification and applies to cryptographic modules only (compared to the more generic nature of CC). Each of the levels defines a certain group of requirements that a cryptographic product has to fulfil in order for the latter to be included in this specific level. The lowest level is Level-1 and the highest is Level-4. Current FIPS version is FIPS 140-2 [32], but it is expected to be replaced by FIPS 140-3 in the future.

2.10 GlobalPlatform

GlobalPlatform is a non-profit organisation that defines a framework under which card issuers can create smart cards that will allow the hosting of multiple applications on them [33]. Application lifecycle is a very important aspect and GlobalPlatform aims at defining a secure way for these applications to execute. The focus is for the security of the smart card to not be compromised. For this purpose, GlobalPlatform defines the notion of ‘Security Domains’. A security domain is a part of the smart card that allows the storage and execution of an application that is provided by the entity that is in possession of the cryptographic keys that can be securely stored on this domain. The application provider signs the application using these cryptographic materials and the application can then execute in this domain only. If someone tries to run another application in this domain, this will fail as the malicious entity will not be able to sign the application with the correct keys (since these keys are handled solely by the legitimate entity that manages this specific domain). The existence of security domains, blocks any application from trying to access another security domain (and thus the resources stored on this domain).

2.11 Smart Cards Security

There are generally three types of attacks against smart cards:

2.11.1 Attacks

2.11.1.1 Fault/Logical Attacks

Software attacks that try to exploit software vulnerabilities/bugs. An example is when an attacker tries to access files on the smart card that have certain permissions [34].

2.11.1.2 Side-channel Attacks

Side-channel attacks are based on data gained from the collection of information of certain physical attributes of the implementation of a smart card-based system [35]. In this case, an attacker can collect information from electromagnetic leaks, power consumption or timing information and can then try to retrieve data stored inside the smart card that otherwise would not have been able to, e.g. encryption/decryption keys. An example is the case where an attacker can get the smart card encryption/decryption key by identifying the difference in time needed for certain commands to execute, e.g. well known attacks against RSA implementations on smart cards where a certain function (squaring) requires/consumes less power to execute compared to another one (multiplication).

2.11.1.3 Hardware/Invasive Attacks

Attacks of this nature are usually very invasive and require direct access with the smart card. For these attacks to succeed, the attacker needs to use a number of tools that will allow her to abuse the card in order to be able to get the necessary information or simply disrupt the smart card execution. One way of doing this is by inserting probes on the smart card to detect all traffic over the bus.

In order for smart card manufacturers to combat these attacks, they have implemented certain countermeasures [18]. A non-exhaustive list follows.

2.11.2 Smart Card Security

2.11.2.1 Secure Architecture - Protective Shields

A protective shield tries to stop physical attacks against the smart card. The shield aims at resisting e.g. probes from being inserted in the smart card and trying to eavesdrop on bus communication.

2.11.2.2 Data Bus Encryption

Another way to avoid bus eavesdropping is by encrypting data that are transferred over the bus.

2.11.2.3 Obfuscation

By obfuscating smart card parts (thus randomising the location e.g. of memory or the processor), smart card manufacturers aim at stopping an attacker from being able to map and identify smart card parts, which then makes it even more difficult for her to launch specialised attacks.

The next section will discuss about the Subscriber Identity Module (SIM) cards.

2.12 Mobile Telephony and Subscriber Identity Module (SIM) Cards

1st generation mobile telephony (called 1G) was analog-based and the first digital network came into existence with the second generation (2G), also called GSM (Global System for Mobile Communications) in Europe.

Early mobile networks (1st and 2nd generation) were vulnerable to a number of security attacks. Some of the most usual were the ease for eavesdropping (1G), as well as insecure cryptographic algorithms (2G). More specifically, the absence of encryption capabilities and the non-existence of a secure module was a major problem with 1st generation mobile telephony. Anyone, with the relevant equipment, could intercept all communication from a target mobile phone. At the same time, the absence of a secure module, meant that there was not a secure way to authenticate the user to the network and phone fraud was relatively easy, since the attackers could seem as the legitimate subscriber and place phone calls charging her instead. 1G networks also suffered from poor quality, regular line drops and inefficient use of the spectrum due to the analog nature of the technology.

2G mobile networks tried to change this. 2G was the first digital mobile network technology. This alone was a big step, but maybe an even bigger step was the existence of a small module/card on the phone, what is called the Universal Integrated Circuit Card (UICC) or more commonly known as Subscriber Identity Module (SIM) card. The UICC is basically the actual card, the hardware, whereas the SIM is an application running on top of it, however in bibliography and everyday use the term SIM card is used to describe both. It must be noted however that SIM cards in 3G and 4G networks are instead called Universal Subscriber Identity Modules (USIM) cards, offering larger storage, richer attributes (e.g. can store email addresses in the contact list) and allows for higher speed connections and support for Internet Protocol (IP) traffic. Throughout this text the term (U)SIM or SIM card will be used interchangeably.

The main purpose of the SIM card is to contain the International Mobile Subscriber Identity (IMSI) and the secret key (Ki). The IMSI is usually a 15-digit number that uniquely identifies the subscriber - no two SIM cards will ever have the same IMSI. The IMSI is extremely secure and sensitive and is rarely published/transmitted in order to mitigate against attacks such as eavesdropping. Instead a randomly generated Temporary Mobile Subscriber Identity (TMSI) is created and used instead. The TMSI is sent every time that the SIM/phone moves to a new geographic location. As mentioned above, the IMSI is rarely being sent and usually this happens when the phone is switched on.

The Ki is a secret key that is stored on the SIM card, but also on the mobile network premises and it is used as part of the authentication process. The authentication process initiates when the phone is switched on and on the mobile network side the Ki, along with the authentication algorithm (A3) and a 128-bit random number (RAND) are used to generate a 32-bit signed response (SRES). At the same time, a ciphering key (Kc) is generated using Ki, RAND and the A8 algorithm and this will later be used to encrypt the communication with the phone.

The RAND is then sent to the mobile device that forwards it to the SIM. The SIM uses the RAND, the Ki and the A3 algorithm to produce the SRES* that sends to the network. The network checks if the SRES is the same as the SRES* and authenticates (or not) the SIM and subsequently, the subscriber. The SIM also calculates the Kc from Ki, RAND and A8. From the above it is obvious that the security of the whole process depends on the security of Ki. Thus, the Ki is protected by being stored on the tamper resistant device that the SIM is, and is not allowed to be read by any entity. It is only used inside the SIM card and never travels outside of it.

From then on the communication can be encrypted using Kc and the encryption algorithm A5. The A5 algorithm creates a 114-bit stream that is then XORed with the data

– this stream changes very often to improve security. In fact there are more than one A5 algorithms, A5/1, A5/2, A5/3, etc. and can be used in cases where a country does not allow a certain type of algorithms or if one of the algorithms is found to be insecure.

Unfortunately, the A5/2 algorithm was proven to be extremely insecure, since it was designed to be used in countries with restrictions on the use of cryptography. Using the A5/2 algorithm is very dangerous since it is quite trivial for an attacker to retrieve the Kc. Combined with a 'false base station' attack, this can be used to retrieve the Kc. The false base station is an attack where a false station can be placed very close to the victim, retrieves information encrypted using the A5/1 algorithm and forces the victim to fall back to the A5/2 algorithm. The false station can then be used to retrieve the Kc and decrypt all previous intercepted communication.

It is easily understood that the security of the entire GSM network is based on the tamper resistance attributes of a SIM card, since a SIM card must be able to impose big obstacles to anyone trying to physically access the Ki or the IMSI. This is the main use of a SIM card and this remains true even for modern mobile networks, i.e. third (3G) and fourth generation (4G).

The SIM card was also used as a medium for the mobile network operator to store and execute programs. Historically, this was done via the SIM application toolkit (SAT) [36]. What the SAT was doing, was to present to the user information about news, the weather, etc. as a response to a request from the user - the user could use a specific application to browse a menu and sees the information on the phone. However, with recent advances on mobile phones technology (smartphones), this functionality is currently not important and in more advanced countries, is practically extinct. For more information on mobile telephony, see [37].

2.13 Conclusion of the Chapter

In this chapter we described the tamper resistant token, the smart card, which its use we are going to be assuming throughout this thesis. We provided a brief history of smart cards, their basic physical characteristics and the most important standards that govern them. We provided an overview of the main components of the smart card, the most important operating systems that reside on a smart card and the certifications that a manufacturer can aim at acquiring for a smart card. Finally we gave a synopsis of the main attacks against smart and mitigation techniques that can be used, before concluding with the representation of the smart card type that we are going to be referencing in this thesis, the SIM card. The next chapter will provide an overview of the World Wide Web and the Smart Card Web Server.

Chapter 3

The Web and the Smart Card Web Server

Contents

In the previous chapter we provided a description of smart cards, before proceeding in this chapter with the presentation of the second important component of this thesis, the Smart Card Web Server (SCWS).

Prior to describing the SCWS, it is important to give an outline of the World Wide Web (Web) and some of the main types of threats against it. These threats are chosen because we will try to mitigate them through the use of a smart card and an SCWS. The latter entity (the SCWS) will be presented in detail in this chapter, before proceeding with the illustration of its use in the next chapters.

3.1 Introduction

When Tim Berners-Lee created the first instance of the World Wide Web, he may have never thought how this will evolve in the future. It is true that the Web dominates a large part of our everyday lives and it has indeed changed the way that society operates.

Unfortunately, although the Web is one of the most exciting inventions of the twentieth century, it brings along a number of challenges. Security concerns are at the top and it is an unfortunate fact that security incidents are very common.

Three of the most common security concerns will be investigated in this thesis. The first is the Denial of Service attacks, the second is the authentication problems on the Web and the Web of Things and the third is about credit/debit card data security and compliance.

3.2 Security

3.2.1 Denial of Service Attacks

A Denial of Service (DoS) attack is any attack that aims at rendering a victim system inaccessible and not able to respond to requests from legitimate entities [38]. The basic idea is to consume as many (or all) of the resources of the victim system - in this case the system will not be able to fulfil requests or its answers will be so slow that it will be practically unusable. In the case of network DoS attacks, the attacker is using only one system against the victim, but there are also attacks aiming at specific vulnerabilities that exist on a system, e.g. an attacker may install a malicious program that at a specific moment will create a large number of processes that will consume the available memory or will start writing on the disk, thus consuming space making the system unusable.

3.2.1.1 DoS Attacks Methods

There are many types of DoS attacks and an indicative number of them is presented below:

1. Ping of Death: This is one of the oldest attacks and there are not many systems that are still vulnerable, as most of the vendors have long ago created patches for this attack [39]. Historically, a ping of death was a very successful attack, affecting a number of systems. This type of attack occurs when an attacking systems sends a malformed, malicious ping to a victim computer (ping is a command sent to detect if the receiving system is live or not). The maximum IPv4 packet including the header is 65535 bytes. The Data Link layer has a limit on the maximum frame size that it can handle, which is 1500 bytes on an Ethernet connection. Thus a large IP packet has to be split in multiple smaller packets (sometimes called fragments) in order for it to reach its destination where all fragments are reassembled. An attacker can manipulate the size of the fragments that she sends to the victim and send a larger overall packet than the one that the receiver can handle. In this instance the victim cannot handle the larger packet and it crashes.

2. Teardrop attacks: Teardrop attack is a DoS attack that is achieved by sending a number of fragmented packets to a victim system [40]. Due to a bug in TCP/IP fragmentation reassembly, the victim machine cannot properly reassemble the receiving packets and thus it crashes. Every IP header includes a fragment offset field that indicates the relative position of the data that the fragmented packet contains against the data in the original packet. If the sum of the offset and the size of data in a fragmented packet are different than the ones of the packet that comes after it, then the packets overlap. This can cause a system crash, as the latter cannot properly reassemble the receiving packets. While this is a quite old attack, a newer version of it affected Windows Vista and Windows 7 [41].

3. Fork bomb: During this attack, a malicious program creates a process on a victim system [42]. This process creates (forks) another process, which creates another process and so forth. Quickly all available resources on the system are consumed and it crashes or becomes slow to respond.

3.2.2 Distributed Denial of Service Attacks

A Distributed Denial of Service (DDoS) attack is a more advanced and catastrophic one, since it involves multiple attacking devices, which is not the case of DoS attacks. In most occurrences, a DDoS is conducted through the use of compromised devices (called zombies) that co-operate to attack a victim device. The owners of these machines may not even know that they are participating in an attack and this is a big risk, since they cannot stop something that they are not aware that is happening. The attacker is controlling this large group of devices (that can be up to hundreds of thousands) that is called a botnet, through the use of a Command and Control Centre. The attacker can then send an initiating command to these zombies that they start sending traffic against the victim device. Depending on the amount of traffic and the defences of the victim, this excessive amount of traffic may or not be absorbed.

3.2.2.1 DDoS Attacks Methods

1. Smurf attacks: A smurf attack, is one of the oldest distributed DoS attacks. During a smurf attack, the attacker sends multiple Internet Control Message Protocol (ICMP) ping requests to multiple recipients. However, it does so as to seem that these pings come from another entity, i.e. the victim system (this is called spoofing). The result is that all systems that received a ping will reply back to the spoofed system that may not be able to cope with all these requests and may crash, slow down or become unresponsive [43].

2. Volumetric attacks (layers 3 and 4): These types of DDoS attacks can be considered a form of brute force, since their main vector of realisation is to flood a target with traffic that saturates the available bandwidth. Usually these attacks are not very sophisticated, but use excessive power to bring down a target. This overloading of the capacity of the victim network forbids access to legitimate users and can bring down a server in a very short period. These attacks target mainly the network level, thus layers 3 and 4 of the TCP/IP protocol stack [44] [45].

3. Volumetric attacks (layer 7): These are more sophisticated attacks compared to the ones mentioned in the previous section. An example of such attack is the HTTP flood where the attacker forces a number of zombies to send POST or GET requests to the victim server. The server tries to reply to all these requests, but if these requests are very resource intensive or originate from multiple sources, then the server cannot cope with all this traffic, causing it to crash or simply slow down or become unusable [44] [45].

3.2.3 IoT-based Attacks

A defining period for these types of attacks can be considered the autumn 2016 where the worst DDoS attacks ever reported, took place. Attacks against the website ‘Krebs on Security’, the French hosting provider OVH and the DNS provider DYN, are attributed to compromised things of the Internet of Things that were infected by the Mirai malware. This malware turned these things into zombies and forced them to participate in huge DDoS attacks [46]. All these incidents were very powerful attacks, with heavy traffic destined against victim systems. Akamai, the company providing

DDoS protection to the ‘Krebs on Security’ website, had to take the website offline, since a full mitigation would have cost millions of dollars. A very big concern with these types of attacks is that they can easily be used by script kiddies that browse the Dark Web and pay for these types of services - this is now called DDoS-as-a-service. Code for Mirai and other malware is also publicly available and even people with limited technical expertise can now use this code to try and infect insecure things, that will subsequently be used as attacking nodes. The entire process is enhanced due to the fact that a vast majority of things are natively insecure [47] and thus very easy targets for infection.

3.2.3.1 DDoS Attacks Mitigation

Mitigating a DDoS attack can occur in the following ways [48]:

1. By extending the internet capacity of the website/system under question. However, this is not always feasible.

2. By using a Domain Name System (DNS)-based cloud service that will permanently forward traffic destined for a specific website to a mitigating service that will then deliver clean traffic, by changing the DNS A-record to point to the mitigation provider’s infrastructure. DNS requests are now resolved by the anti-DDoS DNS servers that are now the authoritative DNS servers for the domain under question.

3. By using a Border Gateway Protocol (BGP) based cloud service that will receive the malicious traffic during the attack and will forward back to the victim the clean traffic. This protection is either ad-hoc and occurs when there is an actual attack or can be always-on. This is called BGP anti-DDoS protection and it is implemented by switching from the victim’s Autonomous System to that of its mitigation provider. The latter can then pass the malicious traffic through its scrubbing service and deliver the clean traffic.

4. Another option is when the anti-DDoS protection is offered directly from the internet service provider. E.g. Level 3 claim that they can avoid DDoS attacks up to 4.5 Tbps [49]. This is an always-on protection and differs from the second option in the sense that no DNS changes are needed, but the traffic is directly filtered on the ISP.

After discussing about DDoS attacks and mitigation techniques, we are now going to present the second threat that affects the security on the Internet and this is about insecure authentication.

3.2.4 Web Authentication

There are two standardised ways to authenticate on the Web:

1. Basic Access Authentication. It is the most primitive way to authenticate on the Web. The client is presented with a pop up window into whichshe enters the authentication credentials (username and password). The credentials are transmitted unencrypted, unless protected by another layer, e.g. TLS. Basic Access Authentication uses Base64 encoding. It is considered as a non secure way for authentication [50].

2. Digest Access Authentication. Digest Access Authentication is an improvement over Basic Access Authentication, in the sense that the username/password pair is not sent in cleartext. Instead an MD5 of the username and password is sent, along with a nonce (that was generated by the server and sent to client) plus a number of other values (e.g. the requested Uniform Resource Identifier(URI)). Again, everything is transmitted over HTTP by default, unless a secure protocol is used, e.g. TLS [50].

3.2.4.1 Form Authentication

In practice web developers and administrators are using non-standardised ways for user authentication. This is usually in the form of an Hypertext Markup Language (HTML) login form/web page. The user is presented with a form that requires a username and password before she is allowed access to protected/private information. This username and password pair is usually selected by the user and specifically for password selection there may not exist a password strength check. This creates the problem that the organisation holds a lot of passwords being insecure. This is further compounded by the use of easy to remember and/or passwords that are used for other websites the user may have accounts on, as the users choose passwords that are easy to remember and/or are used with other websites as well. Password re-use is a big problem for the modern Web and various methods are proposed and/or used to mitigate the risks that arise from the undesireable use of insecure passwords.

Some of the most prominent of these authentication protocols are presented below:

1. OpenID: OpenID [51] is a protocol supported by large organisations according to the OpenID Foundation. Google, Microsoft, PayPal and Amazon are among these organisations.

   In OpenID architecture there is a ‘relying party’ (a website that wants to verify a user identity) and an ‘OpenID provider’ (a website that provides authentication services). The user creates an account with an OpenID provider - this account is then used to authenticate the user. If the user wants to authenticate to a second website that supports OpenID, she does not need an account with the second website. Instead the user authenticates to the OpenID provider, who then transfers the authentication result to the relying party through the user's browser. How the OpenID provider authenticates the users is up to its website's implementation, thus it may be a simple password or use of more advanced techniques like smart cards or biometrics.

2. OAuth: OAuth is an authorisation standard [52]. OAuth allows a user to authorise a website to access some of their information on another website, without having to provide authentication credentials e.g. assuming that a user wants to allow a website to access photos of the user posted/hosted on another website. The user is called the 'resource owner', in the sense that they are the ultimate entity that authorises access to her photos. The authorisation server (i.e. the website containing this photos), allows the other website to access these photos, if the user specifically authorises it. For this purpose the requesting website presents an access token/'valet' key to the authorisation server and gets access to this information, without the user having to submit any credentials to the requesting website.

   It must be noted that OAuth is an authorisation and not an authentication standard. However, in practice certain implementations are using OAuth as a pseudo-authentication protocol, assuming that the user that provides the authorisation has already authenticated with the authorisation server. The use of an authorisation protocol for authentication is a security risk (as these are two different attributes). This risk was dealt with OpenID Connect.

3. OpenID Connect: OpenID Connect [53] is an authentication layer that sits on top of OAuth and adds authentication information to the authorisation information. The protocol allows for third-parties to request and obtain authentication information for a user, in addition to access to information that the user allows. OpenID Connect thus combines both 'worlds' of OAuth and OpenID.

4. SAML: The Security Assertion Markup Language (SAML) [54] is a protocol that allows for authentication and authorisation information to be exchanged between parties. It is XML-based and its specification is provided by the OASIS Security Services Technical Committee.

   SAML's main use is for web browser single sign-on (SSO). The 'principal' (i.e. the user), the 'identity provider' (IdP) and the 'service provider' (SP) are the three main entities in the protocol. The user authenticates with the IdP and presents the authentication result to the SP. This result is in the form an identity assertion - depending on the implementation this assertion may also contain authorisation information and not only authentication. The SP can then accept or reject the assertion and allow the user to access resources offered by the SP. Usually between the SP and the IdP there is a pre-existing establishment of trust that occurs through the exchange of the necessary X.509 certificates.

The method of authentication is left to the implementation, i.e. it may be a password, a biometric or multi-factor.

In reality, SAML is used exclusively for corporate applications. In this instance, the organisation's Active Directory or LDAP directory usually provides the identity and the service providers are web applications that the user is authorised to use under a corporate identity. Usually an intermediary is used in the form of a paid service like Okta or OneLogin that allows for easy and secure connection between the identity provider and the service provider. This intermediary plays the role of the identity provider.

The above authentication methods aim at facilitating the way that users authenticate on the Web, but security problems still exist.

3.2.5 Phishing Attacks

One of the biggest threats on the Web is how users authenticate to web applications [55]. It is a reality for a large number of web users that they are required to remember and maintain a large number of passwords, that have to be unique for each website. Additionally, these passwords have to be strong enough to avoid being guessed by attackers.

In practice, these good practices are followed only from a minority of users. The vast majority are reusing passwords across multiple websites and the even biggest risk is that many of them are not able to identify malicious, phishing emails. On top of this, large hacks, like the ones that occured with Yahoo [56] leave users exposed even when they are indeed following all proper security guidelines.

3.2.5.1 Phishing and Antiphishing Mechanisms

Phishing is the most prominent and simplest to implement method for attackers to steal authentication credentials from users. Phishing is any attempt aiming at surreptitiously obtaining a user's username and password credentials by tricking the user. Attackers are using legitimate-looking (but under the attacker's control) websites that convince the user that the website is indeed legitimate. Once the user submits the credentials, these are then received by the crook, who in turn is able to use them and authenticate as the user.

To mitigate phishing attacks, the industry developed a number of solutions e.g. anti-spoofing mechanisms are employed by all security aware companies/organisations, while at the same time anti-impersonation tools aim at detecting phishing emails at the perimeter of the corporate networks (by using a number of parameters, like the time that a certain domain is active, how similar is with the corporate domain, etc.). The disadvantage is that these solutions are for a corporate environment and not the end user and even on corporate networks they are not always successful at mitigating the threat.

On the other hand, security awareness could be considered the most important security solution againt phishing attacks. Instead of putting in place just technical solutions, a security training tries to educate the end user to be able to detect potential phishing attacks. However, there have been instances where even well trained and security-aware users fell victims to a phishing attack [57].

Another mitigating technique comes in the form of a picture that the user chooses upon registration. This picture must be shown on the login page and if it is not, then the user will understand that the website is not legitimate (since the malicious website will not know which is the correct image). Unfortunately, even with this solution, users are ready to submit their credentials, even if they do not see the login image.

The result is that users are constantly falling victim to phishing attacks.

A solution to the problem is to use what is called a second-factor of authentication, i.e. apart from the username and password pair, the user is required to submit a second value (usually a one time password, i.e. a password that is valid for a short period only). This second factor comes in a variety of forms [11]. It may be an SMS with a code, an application running on the phone that provides one-time passwords (e.g. Google Authenticator) or a hardware token, e.g. a Yubikey. However, the former solution is no longer considered adequate as advised by the National Institute of Standards and Technology (NIST) [58]. Similarly the Yubikey is not widely supported at time of writing and it also has a monetery value (as with all other hardware tokens). Google Authenticator (and other similar solutions like Microsoft's Authenticator) is a security solution without known vulnerabilities so far, but it assumes that the user's phone is secure. Once an attacker gets hold of the phone, there is nothing to stop her from using Google Authenticator to authenticate to websites. Another important factor that needs to be considered is that Google Authenticator runs on the insecure part of the phone (Android OS) and there is no guarantee that there will not be attacks against this application. Additionally, from a usability point of view, some users may find it difficult to install the respective application and/or use it.

Apart from these authentication concerns, another big problem on the Web is the security of e-commerce transactions.

3.2.6 Credit/Debit Card Data Security Problems

With the big advancement of electronic commerce, the necessity for credit/debit card use on the web became a reality. A customer must be able to pay in an easy and safe way as it happens when in store. These transactions are called card-not-present (CNP), as the physical card is not in proximity to the merchant's Point of Sale (PoS) equipment as opposed to card-present (CP) transactions where the actual card is presented and used inside the shop. However, the biggest concern with CNP transactions is exactly the fact that the merchant cannot confirm that the person that purchases the goods is the real owner, whereas with CP transactions there are countermeasures. With the EMV chip cards for example, a PIN is needed to authorise the purchase. These cards also employ certain other security characteristics. Another example is the predecessor to the EMV chip card where the retailer can check the purchasers identity by asking for an identity card or a signature to match against the one on the reverse of the card. For CNPs to be used in a more secure way, the card is delivered to the client with a three digit code at the back of the card. This code is called Card Verification Value (CVV). When the customer wishes to purchase something on the Internet, she will be asked to provide that number. This provides some assurance to the merchant that the buyer is indeed the real owner of the card or at least has the card in their possession.

In practice this has proven to be partially effective. A number of security breaches resulted in the exposure of credit/debit card numbers with the CVVs that accompany them. This practically means that the security that the CVV offers is compromised [59].

To combat the problem of credit card fraud, the leading credit card companies (including Visa and MasterCard) created a security standard that must be followed by anyone that processes credit/debit card data. The standard is the Payment Card Industry Data Security Standard (PCI DSS) [60]. PCI DSS is now in version 3.2, is a reasonably developed standard and its effect is felt since it was initially launched.

The effect that PCI DSS has brought to the battle against credit/debit card fraud is still questioned by a number of experts [13], while others keep a more positive stance. However, what is generally agreed is the fact that compliance with PCI DSS is difficult and expensive to maintain. Becoming PCI DSS compliant assumes a significant investment in IT infrastructure, process changes and human capital that may not be affordable to every company.

Note: IoT/WoT security concerns will be mentioned in Chapter 7.

After the introduction of the areas that will be investigated in this thesis, we will introduce the basic entity that we are going to use in order to tackle the security concerns in the above areas, the Smart Card Web Server.

3.3 Introduction to the Smart Card Web Server

The Smart Card Web Server (SCWS) is a tiny, stripped down HTTP web server, specially designed to operate in constrained environments as is a smart card. The basic focus of the SCWS is to allow Mobile Network Operators (MNOs) the ability to offer services over 'web' protocols.

The main purpose of the SCWS is to offer services to the user over the browser that is running on the mobile device. Traditionally these types of services were offered by the SIM toolkit, but the emergence of modern mobile devices, that can provide rich content, rendered the SIM toolkit obsolete. Present day users are more demanding, expecting graphics and extensive functionality that will exploit the power of modern mobile devices.

At the same time, users are accustomed to the use of the Web its ability to deliver services be delivered via a web browser is very important. The SCWS is able to offer this as well, since it is a web entity, able to offer its content directly to a web browser.

The SCWS will be able to serve static, as well as dynamic content. Applications register to the SCWS and are identified by a specific URL. When the user tries to access such a URL, the SCWS triggers the application in question, which creates the content and sends it back to the SCWS. The SCWS then sends the content to the user's browser. Such applications are content providing applications.

Before serving the content, the application can present an authentication form to the user, which means that the SCWS can support content that needs a more advanced level of authorisation in order to be accessed.

Another category of SCWS applications is one where they are registered as interception applications. These applications exist passively on the SCWS meaning that they never reply to requests. These applications are triggered before the content providing applications and are mainly used for purposes like logging. This thesis investigates the use of content providing applications. For information about the SCWS, see [61] [62] [63] [64].

Modes of Operation

The SCWS is designed to operate in two modes:

1. In server mode

2. In client mode

The SCWS is said to be operating in server mode when it answers to requests by a client (e.g. browser running on the phone). This can be considered the 'normal' operation for the SCWS. The client mode is when the SCWS is itself functioning as a client. This is the case when the the MNO is remotely administering the SCWS (see below the relevant section).
Figures 3.1 and 3.2 represent both modes of operation for the SCWS.

3.4 SCWS Uniform Resource Locator (URL)

The SCWS, as any other web server, will be accessed through traditional HTTP Uniform Resource Locator (URL) of at least 1024 characters.

3.5 Bearer Independent Protocol Transport Protocol

The Bearer Independent Protocol (BIP) as defined in [24], is used in the case where the card cannot support direct access over HTTPS, i.e. does not implement a TCP/IP stack. In this instance, all communication between the SCWS and the user's browser is happening using the BIP that encapsulates all HTTPS traffic. For the exchange of information to occur properly, there must be a gateway on the phone that will act as an intermediate between the browser and the SCWS.

The BIP gateway (as it is called) will receive HTTPS requests from the browser and will encapsulate these requests in a packet and then will pass it over to the SCWS. The SCWS will then strip the encapsulated packet and will retrieve the original request. The reply will be done in the same way, but towards the opposite direction. Again the BIP gateway will act as the intermediary and translator between the two.

The port numbers that are registered for this communication will be 3516 when communication is over HTTP and 4116 when it is over HTTPS. Also, the IP address used for the communication is the 'loopback' or localhost IP address of 127.0.0.1.

The specification also allows the SCWS to open multiple communication channels with applications on the phone, meaning that the SCWS will be able to serve multiple requests.

3.6 Using the TCP/IP Protocol

If the smart card supports a TCP/IP stack and is able to directly reply and serve HTTPS requests from the phone's browser, then there is no need for the BIP gate-

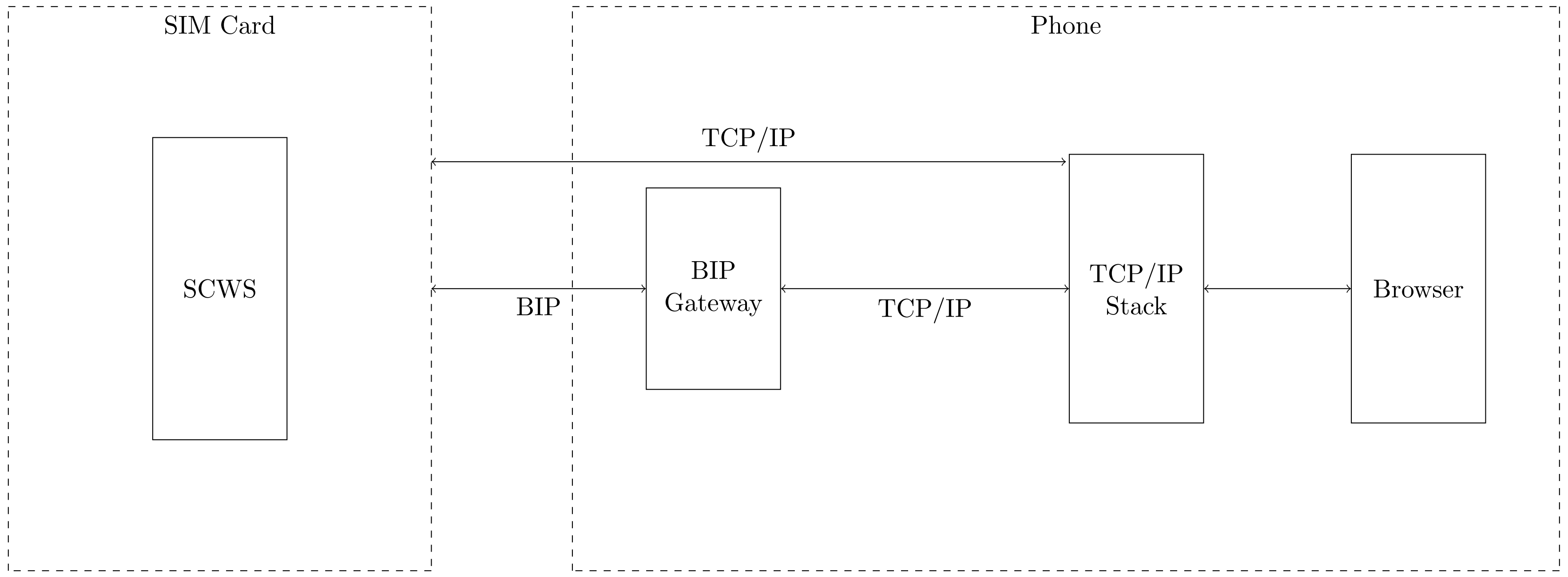


Figure 3.1: SCWS in Server Mode (over BIP or TCP/IP)

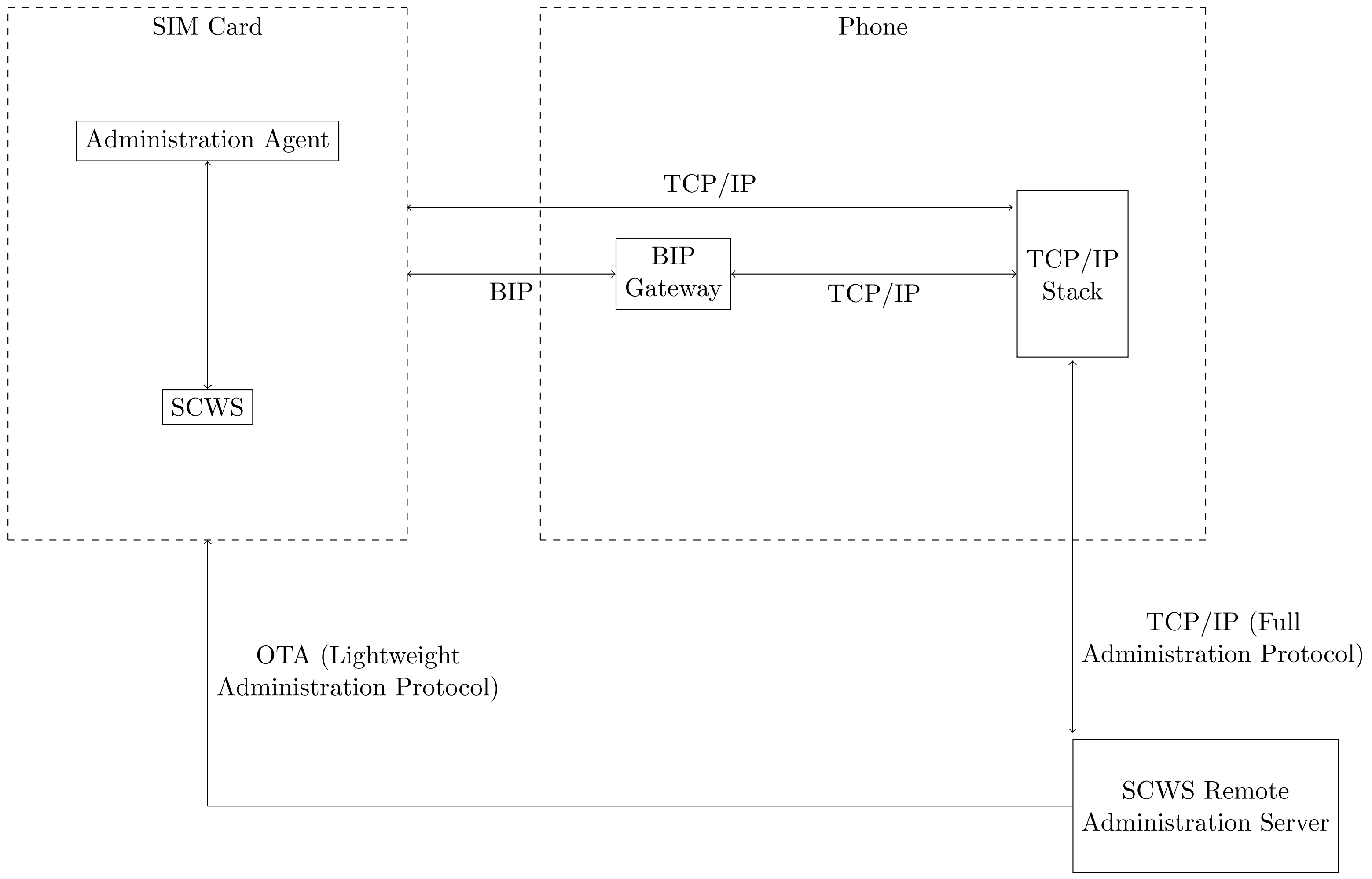


Figure 3.2: SCWS in Client Mode (over BIP or TCP/IP) and Used Protocols

way [25] and [26]. In this case, the port for HTTP communication will be the traditional 80 port, while the equivalent for HTTPS will be the 443. The IP address can be dynamically allocated, but the specification mentions that the card must be addressed by the name ‘localuicc’.

3.7 User or Principal Authentication to the SCWS

The specification requires that the SCWS must support Basic Authentication as defined in IETF RFC 2617 [50] and may also support Digest Access Authentication. However, the triggered application can implement its own authentication scheme (as mentioned above), for example an application specific user name and password or a PIN. This functionality is being used throughout this thesis.

3.8 Transport Layer Security (TLS)

The SCWS must support both ways for TLS communication. Thus, it must support symmetric encryption using PSK-TLS where the communication between the SCWS and any outside entity will be protected through the use of a pre-shared key. The specification requires the use of modern symmetric algorithms, i.e. AES or 3DES.

It must also support asymmetric encryption through the use of public key cryptography and certificates. The specifications requires the use of RSA as a minimum.

3.9 Access Control Policy

The specification also defines another security feature that will protect the SCWS from unauthorised access from applications that run on the phone. It is called Access Control Policy Enforcer and will control which applications have the right to access the SCWS or not. This is particularly important because with current technology, users are allowed to download and install applications on their phones. Some of them may be malicious, so the SCWS must be properly protected from being accessed by rogue applications.

The Enforcer will allow only certain trusted applications to access the SCWS. These applications may be applications provided by the MNO, the handset manufacturer, that are identified by a proper certificate or some other enterprise trust (maybe applications that run inside a trusted environment as provided by Mobile Device Management solutions). The specification mentions that this Access Control Policy will be provided to the Enforcer by the smart card over HTTP.

3.10 Remote Administration

As mentioned above, the SCWS will not only act as a server replying to client requests, but will also function in a client mode when the MNO wishes to administer the SCWS. This may happen in cases where the MNO wants to upload new data e.g. a photo or a new xHTML page, delete data or change configuration parameters.

When the Remote Administration Server wants to start an administrative session, it first sends a Push message to the SCWS administration agent. This message can be through either a normal Short Message Service (SMS) or the Open Mobile Alliance (OMA) Session Initiation Protocol (SIP) Push Enabler. Upon receipt of the administration message, the administration agent initiates the administration session.

There may be certain pages and URLs that will be accessible only to the administration entity (e.g. /SCWS/admin), which means that unauthorised entities will not be able to alter information on the SCWS.
The administration commands are as follows:

1. PUT (is used to install or update a page)
2. DELETE (is used to delete a page)
3. POST (is used to send administration commands to the SCWS)
4. GET (is used when the remote administration entity wishes to retrieve a page or audit the SCWS)
5. HEAD (is used to retrieve the header of a page)

3.11 Administration Protocols

The specification defines two different types of administration methods:

3.11.1 Lightweight Administration Protocol

When the administrator wishes to change a small amount of parameters, then the most appropriate way to administer the SCWS is through the use of the Lightweight Administration Protocol.
This protocol uses current Over-The-Air (OTA) protocols and the commands are delivered through SMS messages.

3.11.2 Full Administration Protocol

When the administrator wishes to make a larger number of changes on the SCWS, then the Lightweight Protocol is not appropriate. In this instance, the Full Administration

Protocol (FAP) must be used. This is the case when pages or a page need(s) to be uploaded to the SCWS or a large number of configuration changes must occur or data must be exchanged with applications residing on the SCWS,e.g. a large image needs to be uploaded on the SCWS.
The full administration protocol is happening over TCP/IP (and eventually over BIP if the card does not natively support TCP/IP) and TLS (for security) and assumes the use of an on-card entity, called the Card Administration Agent.

Card Administration Agent

The Card Administration Agent is an HTTP/1.1 client that acts as the intermediate between the Remote Administration Server and the SCWS.

It manages connection establishment between the Remote Administration Server and the SCWS, retries and reconnects when a connection drops and can be triggered either by external events (communication starting from the Remote Administration Server) or from internal events generated by the card.

Full administration over BIP

When the card cannot support direct communication over TCP/IP, then the card administration agent will communicate with the BIP gateway over BIP commands [24].

Full administration over TCP/IP

If the smart card implements a TCP/IP stack and can communicate natively over TCP/IP, then the card administration agent does not need to encapsulate TCP/IP traffic in BIP commands and can contact directly the remote administration server without the need for the BIP gateway.

It must be noted, that the Remote Administration Server can trigger an administration session either through an SMS sent to the agent or through on OMA SIP Push message.

3.12 User Self-Care Administration

The specification includes a mention that the SCWS may provide the necessary capability to the end user, for the latter to update her password. As mentioned above, this is an expected functionality, in case there is protected content that the SCWS will have to serve to the end user.

3.13 Conclusion of the Chapter

In this chapter we offered an overview of the Web and some of the major threats that the Web is facing. These threats are the ones that in the next chapters we will try to mitigate through the use of the SCWS. The latter entity, was also described, since the reader should understand the main SCWS components before proceeding to the solutions represented later. The first of these solutions will be outlined in the next chapter. The feasibility of using the SCWS as a mitigating factor against DDoS attacks will be described. The solution aims at making extremely difficult and expensive any DDoS attack that can be launched against a voting infrastructure, by distributing the functionality to 'miniature' voting websites that are hosted on the local SCWS for each of the voters.

Chapter 4

Distributed E-Voting Using the Smart Card Web Server

Contents

This chapter is based on the papers *Distributed E-Voting Using the Smart Card Web Server*, 7th International Conference on Risks and Security of Internet and Systems (CRiSIS 2012), Cork, Ireland, October 2012 and *Remote E-Voting Using the Smart Card Web Server*, CRiSIS 2012 special edition of International Journal of Secure Software Engineering (IJSSE), Vol.5(1), IGI Global, 2014.

In the previous chapter we described the main components of the Web and the SCWS, before proceeding in this chapter to present the novel idea of using the SCWS as a means to support a distributed way of conducting elections. This solution allow the mitigation of fears of DDoS attacks against voting websites that aim at disabling or disrupting the voting procedure. The solution is using secure, standardised, well-established protocols like HTTPS. The description considers a voting scheme, named Prêt à Voter, which is a solution in the e-voting field. In this chapter, we will explain how we can use this scheme with a Smart Card Web Server. A security analysis will also be presented, before concluding the chapter.

4.1 Introduction

Voting in elections is a fundamental democratic right, but it can be a challenge to engage citizens and encourage them to vote. Public engagement with the democratic process could be improved by using remote e-voting systems, where a voter uses their own computer or mobile device to cast votes over the Internet. Examples of practical implementations of remote e-voting include elections in Estonia [65] and Switzerland [66]. The fundamental requirements for any voting system are that votes should be recorded as cast, counted as recorded and not linked to a specific voter. Only eligible voters should be allowed to vote, and they can only cast one vote each [67]. Some e-voting systems are designed to address these requirements in the controlled environment of an election poll-site. Examples include fully electronic systems such as Votebox [68], Direct Recording Electronic (DRE) machines [69, 70], and paper-based ballots such as Prêt à Voter [71] and the Scratch Card voting system [72].

Remote e-voting systems, however, have to operate in unsupervised environments, leading to opportunities for denial of service and technical attacks on the voting infrastructure. For example, the voter's equipment could be infected with malware that tampers with the vote, or a Voting Authority (VA)'s centralised web servers could be attacked, as seen in the 2010 Washington D.C. election [73] and the 2012 Canadian New Democratic Party Elections [74]. These attacks can seriously undermine the credibility of an election. In the Washington D.C. case, the e-voting system was broken into within 48 hours of it becoming available, and by taking control of the election server, the attackers 'changed every vote and revealed almost every secret ballot' [73]. Coercion and vote-buying are also problems for remote e-voting. Anti-coercion measures are included in some e-voting systems e.g. Civitas [75], but some schemes are specifically designed for use in low-coercion elections, such as Helios [76].

Although many e-voting processes can be cryptographically protected to ensure the integrity and confidentiality of the votes cast, Rivest [77] identified a critical problem with remote implementations, i.e. 'interfacing the voter to the cryptography'. Security weaknesses in hardware, operating systems and software mean that equipment cannot be trusted. This is known as 'the secure platform problem'. Several methods to address this have been proposed [78]. One approach is code voting, when voting authorisation codes are sent to voters before the election, via a second channel such as the postal service: see [72, 79, 80, 81] for examples.

Two methods could be used to make attacking a remote e-voting system less attractive. Firstly, installing vote processing in a trusted tamper resistant environment that can only be accessed by authorised parties will reduce the opportunity for malicious modifications to the voting application. Secondly, distributing vote processing over a large number of web servers will mean that an attacker must target multiple sites to be successful. The most ubiquitous tamper resistant security token available is the mobile phone SIM card [17]. A SIM equipped with a SCWS, has web server functionality in the SIM environment, so a distributed vote processing application can therefore be run in a tamper resistant environment.

A mobile phone SIM is a restricted processing platform, however, so a voting application cannot necessarily perform all required e-voting system functions. It can provide a 'front end' input method to more sophisticated cryptographic e-voting systems which do have the required resources. This chapter proposes a generic SCWS-based voting procedure that can be used with existing e-voting systems to provide a complete voting solution. As an example, the SCWS approach will be shown with the e-voting system Prêt à Voter (PAV) [67], as this scheme has features which can be readily adapted to work on a phone.

This chapter acknowledges the problems caused by coercion and vote buying in remote e-voting, but does not attempt to offer a solution. The purpose of the proposal is to minimise the potential for web server based attacks on an e-voting system by moving this processing to the tamper resistant SIM, whilst relying on the MNO's trusted SCWS administration protocols to transport the votes to the VA. The chapter is structured as follows: notation used is shown in Table 4.2. A generic model for using the SCWS in a voting system is presented in Section 4.2. The e-voting system, Prêt à Voter, is briefly described in Section 4.3, before being used as an example with the SCWS generic model in Section 4.4 to provide a complete remote e-voting system. A preliminary security analysis of the proposal is presented in Section 4.5. The Scyther tool is presented in Section 4.6 and the formal analysis of the protocol that it provides in Section 4.7, before the chapter concludes in Section 4.8.

Table 4.1: Security Credentials

Credential	**Description**	**Location**
Voter ID	Size and complexity according to VA's policies - it is better to avoid having a voter ID similar to the voter's surname or other demographic information, so that it will be more difficult for an attacker to guess.	Stored on the SIM. Known by voter and VA.
Password	Size and complexity according to best practices [82] - has to be at least 8 characters long with upper/lowercase letters, at least a number and a punctuation character.	Stored on the SIM. Known by voter and VA.
Key Pair (public/private)	A cryptographic key pair (PUB_V, PK_V) used for encryption/decryption, signing/verifying - for key separation purposes, it is better to have a different key pair for encryption and another one for digitally signing (but this depends on the VA). The key size should be according to best practices (see NIST SP800-57 Part1 [83] for details).	Private key stored on the SIM alone. Public key also known to the VA.
Public Key (VA)	PUB_{VA} will be used to encrypt data sent to VA by the SCWS (e.g. the vote). It is best practice to use a different key for every election, so that brute force attacks against it (if successful) only affect one election .	Stored on the SIM and known to everybody

4.2 Generic Model for E-Voting Using SCWS

4.2.1 Entities and Assumptions

The entities involved in the proposed generic e-voting system are as follows:

- Voting Authority (VA): The VA has details of all voter credentials, candidate information, ballot forms and election parameters. It is responsible for creating the voting application, registering voters, and receiving and counting the votes once they have been cast.
- Smart Card Web Server (SCWS): as described previously.
- Remote Administration Server: This is the interface between the e-voting system and the voter's mobile SCWS. It updates the SCWS of a voter's registered phone via the MNO's Full Administration Protocol, using HTTPS.
- Mobile Network Operator (MNO): The MNO provides management services such as Lightweight and Full Administration Protocols to update the SCWS on the SIM, and a mobile network infrastructure.
- Voting Application: Installed on the SCWS (as described in [33]), this displays ballot forms and collects votes ready for transfer to the VA.
- Mobile Phone and Browser: The mobile phone handset and browser application are generally untrusted, as 'jail-broken' operating systems or malware can compromise the correct operation of the phone's applications and operating system.

It is assumed that:

- For the purposes of the proposed design, the mobile phone has a one-to-one mapping to the voter, allowing just one vote per SCWS.
- The VA must be trusted not to collude with the MNO, as the MNO only provides management procedures and infrastructure and should not be trusted with sensitive voter, ballot form or election information.
- A secure voter registration procedure is in place. The exact details are out of the scope of this chapter, but a voter must supply a mobile phone number to the VA for e-voting via a phone. It is assumed that voter credentials (voter ID/password) will be generated in a secure way, along with a voter public/private cryptographic key pair (PUB_V, PK_V) to be used in the voting process. The public key of the VA PUB_{VA} will also be required: the security credentials which are assumed to be available following registration are shown in Table 4.1.

Table 4.2: E-Voting Notation

Notation	**Description**
PUB_{VA}	Public Key of the Voting Authority
PUB_V	Public (Encryption) Key of the Voter
PK_V	Private (Decryption) Key of the Voter

4.2.2 Installation of Application onto SCWS

The data shown in Table 4.1 is sent to the voter's SCWS by the remote administration server, using the Full Administration Protocol. Firstly, (PUB_V,PK_V) and PUB_{VA} are sent to the SCWS. After this procedure has completed successfully, the voter ID / password and application (code/data) are transferred in the same way, encrypted by PUB_V for decryption on the SIM using PK_V. The communication channel in both occasions is protected by HTTPS. This is shown as messages 1 and 2 in the protocol diagram in Figure 4.3.

The voting application (a Java servlet running on the SCWS) must be given access to the voter credentials and the keys installed on the SIM, and is able to create dynamic content [61]. The application creates this content whenever requested and returns it back to the SCWS (which in turns serves this content to the voter as HTML pages). The voter may be notified about the installation, for example via an SMS, an automatic call (such as an Interactive Voice Response (IVR)), an e-mail to a pre-registered e-mail address or by simply updating the SCWS home page with a link pointing to the voting site inside the SIM card. This is shown as message 3 in the protocol diagram in Figure 4.3.

Note: it is generally considered a better practice to avoid transmitting private keys. Keeping these keys secure in a tamper resistant environment where they were already created, means that these keys remain reasonably secure for their entire life time. For a private key to be transferred from the location of its creation to another location, there must exist at least three main conditions (among others):

- The sender and the recipient must have absolute trust in each other.
- The recipient must offer at least the same level of security as the one that the sender can offer.
- The communication channel is at least as secure as the information that has to pass through it.

It is expected that all above conditions will be met. However, to avoid going through this whole process, the Voting Authority can have already in place a mechanism to provide SIM cards with pre-installed public/private key-pairs or to allow them to be created on the SIM as per request. In this thesis we are not going to consider such option and we will assume that the whole process mets at least the above three criteria.

4.2.3 Authentication

The VA will again contact the voter through an SMS, IVR call or e-mail when the election starts, and the SCWS homepage will show a 'VOTE NOW' link . Clicking the link transfers the voter to the SCWS environment and an authentication page is displayed. In this page the voter will enter the voter ID/password that were issued during the electoral registration. The voter ID/password are checked against the ones that were previously transferred to the SIM card, and that the voter ID has not been used to vote already. If the authentication is successful, the voter is presented with the ballot form, in a format determined by the e-voting system used. Again the communication takes place over HTTPs, shown in messages 4 to 7 in Figure 4.3.

4.2.4 Choosing a Candidate

On the ballot form, next to each candidate there is a radio button that the voter has to check in order for the candidate to be chosen. Once the choice has been made, selecting a 'VOTE' button generates the next screen, asking for confirmation of the candidate chosen (it is possible to include a text entry option for code voting schemes). Once the voter confirms, the vote is submitted to the SCWS. These are messages 8 to 11 in the protocol diagram in Figure 4.3, and again all communication is over HTTPS. At this stage it is possible to include a dummy vote (if the e-voting system allows this for verification purposes, e.g. in [84]).

4.2.5 Vote Storage and Sending

The vote is now stored on the SIM card. The format of the vote is dependent on the voting scheme used: for example, it could be a pre-assigned vote code as in [79], or a cryptogram which determines the position of the candidates on the ballot form as in [67]. The vote is encrypted with PUB_{VA}, and this will be retrieved by the VA and subsequently deleted from the SCWS after it is cast. The vote is also kept on the SIM for future reference, encrypted with PUB_V. This encryption is not strictly necessary, but is included to provide an additional security measure in case space restrictions mean the vote is stored in non-tamper resistant memory or on the phone. A flag is set to indicate that the voter ID has been used to vote. Once the vote is encrypted, the SCWS administration agent will either trigger a connection with the remote server, or the remote server can automatically retrieve the vote on a specific day/time. In both cases the two entities initiate a HTTPS connection and the remote server retrieves the vote through the use of the Full Administration Protocol. Once the vote is received by the VA, the voter receives a notification that their vote was cast via a second channel

Name	Vote	Code
Bob		
Charlie		
Alice		
		127645

Vote	Code
X	3672
	127645

Figure 4.1: Prêt à Voter Ballot Forms Before and After Voting

e.g. an SMS message, a voice message or an email. Steps 12 to 15 in Figure 4.3 show the vote storage and sending process.

We now turn our attention to Prêt à Voter, an example of an e-voting system which could be used with the generic model of SCWS vote processing.

4.3 The Prêt à Voter E-Voting System

Prêt à Voter (PAV) [67, 84, 85, 86] is a paper-based e-voting system designed for the supervised environment of an election polling-station. The paper ballot form is used as input to a cryptographic system. The electoral process can be audited by voters and third parties, to give end-to-end verifiability. Voters can verify the system by performing dummy votes, which are not included in the final election count.

Voters are given a paper ballot form, and they mark their choice with an X. Every ballot form is different, as candidates are listed in a (different) random order on each one. There is a code (called an 'onion') which contains details of the candidate order in encrypted form. When the vote is cast, the left hand side of the form (containing the candidate names) is detached and destroyed, leaving the voter with a voting slip showing the position of their vote on the form, but not who the chosen candidate was.

This is input to the PAV system and the voter is given the slip as a receipt which can be checked against a web bulletin board at a later stage. The actual vote recorded in the system is the numerical position of the voter's choice and the onion i.e. (*index, onion*).

In the PAV scheme, ballot forms can be printed on demand by the voting booth, as described in [85]. Briefly, two related onions are calculated for each ballot form, the 'booth' onion (left onion) encrypted with the public key of the voting booth, and the 'registrar onion' (right onion) encrypted in the normal PAV way. These 'proto-ballot' forms can be distributed and stored securely prior to the election. Alternatively, the left onion could be encrypted with PUB_V (rather than a booth key) without losing any security, by using ElGamal for distributed blinding as described in [87]. On election day, when a voter takes a ballot form into the voting booth, the booth reads the left onion, uses its previously stored secret key to decrypt the onion, reconstructs the candidate list and prints the ballot form. The vote is cast using the right onion, i.e. (*index, right onion*).

In [86], confirmation codes for each candidate are printed on the ballot form. These codes are calculated by the voting authority prior to the election. Once a voter has cast their vote, the confirmation codes are recalculated by the voting authority and relayed back to the voter at the poll-site. The voter can check this received value against a confirmation code hidden on the ballot form under a scratch-off strip, which only they should see. The voter's receipt contains the position (index) of their chosen candidate, the associated confirmation code and the right onion. Figure 4.1 shows ballot forms with confirmation codes, before and after voting.

It is the ability to print ballot forms on demand, along with the use of confirmation codes that make Prêt à Voter a suitable example e-voting system to use with the SCWS model. The next section describes how this will work.

4.4 Using the SCWS With Prêt à Voter

The advantage of using Prêt à Voter with confirmation codes is that most of the cryptography (i.e. generating the onions) is done by the VA before the election starts, so this minimises the amount of cryptographic processing necessary in the restricted environment of the SIM. With suitable modifications to the generic model, the SCWS voting application can play the part of the PAV voting booth, to decrypt the candidate ordering and hence display ballot forms. The SCWS can also store confirmation codes securely so that when the voter receives a code from the VA (in an SMS) the two values can be compared. Figure 4.2 shows an overview of the design.

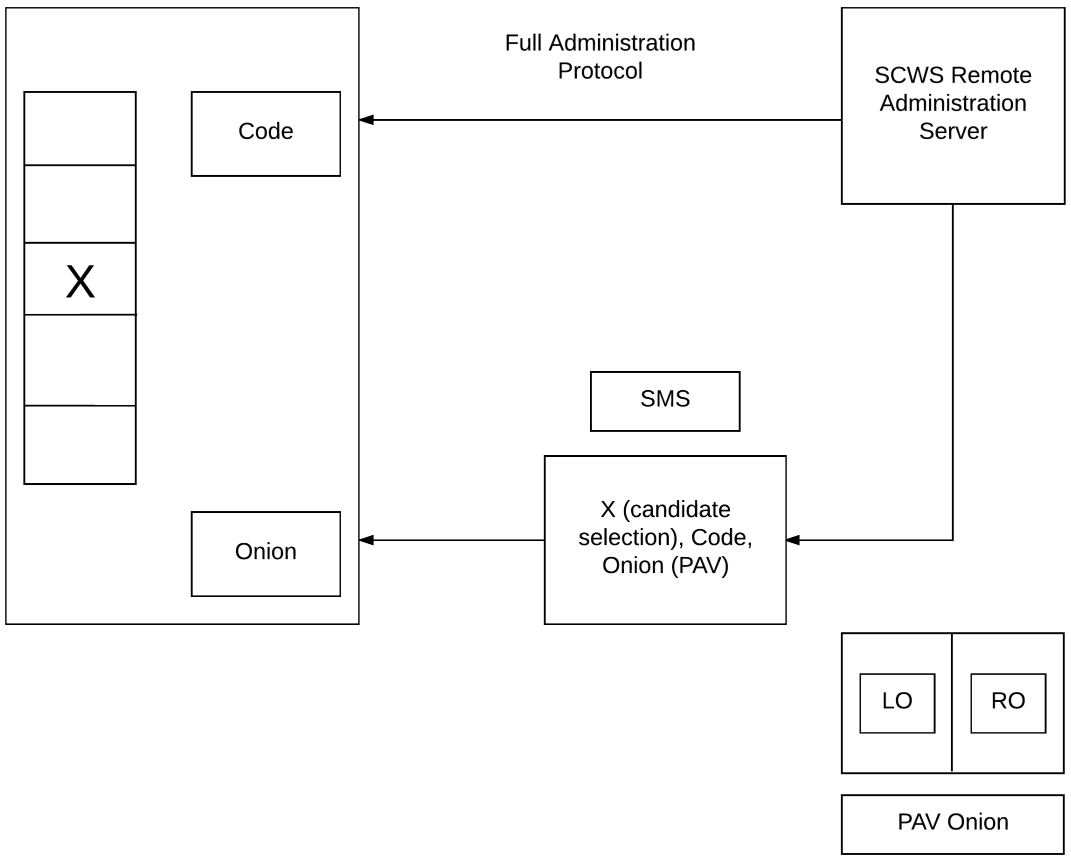

Full Administration Protocol
SCWS Remote Administration Server
Code
X
SMS
Onion
X (candidate selection), Code, Onion (PAV)
LO
RO
PAV Onion

Figure 4.2: Prêt à Voter and SCWS

In message 2 of the protocol shown in Figure 4.3, PAV candidate lists and 'proto-ballots' (including confirmation codes) are sent to the SCWS along with the voting application and other credentials. The left onion must be encrypted with PUB_V, using the El-Gamal blinding technique mentioned earlier. It is suggested that several proto-ballots are sent, as this allows the voter to cast dummy votes to audit the system if they so desire. The authentication stage remains the same as the generic SCWS approach, but when the ballot form is to be displayed (message 7 in Figure 4.3), the PAV method of constructing the candidate order must be followed. This involves decrypting the left onion using ElGamal and PK_V, and then reconstructing the candidate order using a predetermined PAV procedure [84, 85]. The ballot form can then be displayed.

Choosing the candidate and voting is the same as in the generic SCWS approach. Once the candidate has been chosen, the vote needs to be signed with the voter's signing key before it is retrieved by the remote administration server. As the added precaution mentioned previously, a copy of the vote with its associated confirmation code is encrypted with PUB_V and stored on the SCWS. Storing the confirmation code on the SCWS ensures that at a later stage, only the voter can see this code (as specified

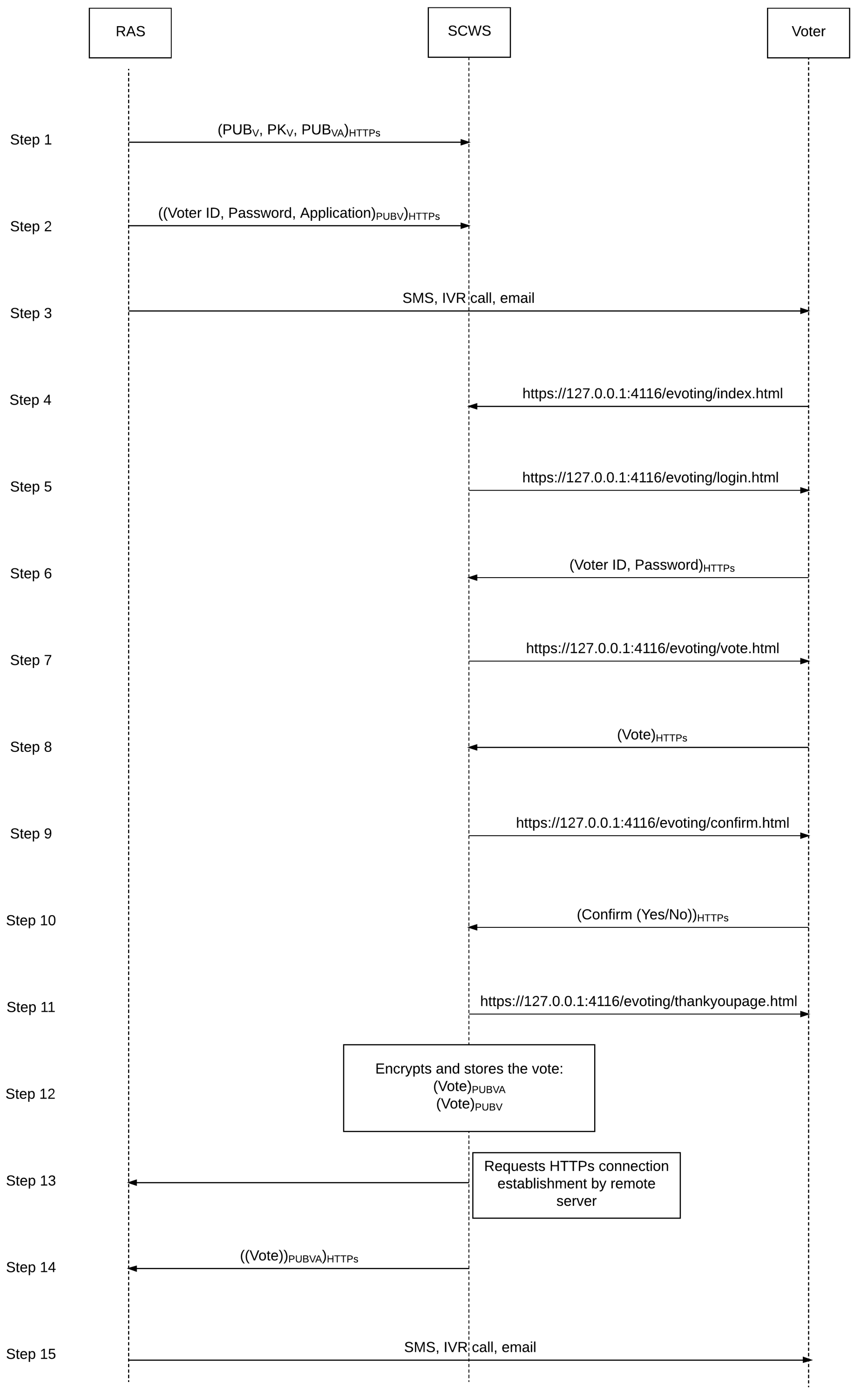

RAS
SCWS
Voter
Step 1
$(PUB_V, PK_V, PUB_{VA})_{HTTPs}$
Step 2
$((\text{Voter ID, Password, Application})_{PUBV})_{HTTPs}$
Step 3
SMS, IVR call, email
Step 4
https://127.0.0.1:4116/evoting/index.html
Step 5
https://127.0.0.1:4116/evoting/login.html
Step 6
$(\text{Voter ID, Password})_{HTTPs}$
Step 7
https://127.0.0.1:4116/evoting/vote.html
Step 8
$(\text{Vote})_{HTTPs}$
Step 9
https://127.0.0.1:4116/evoting/confirm.html
Step 10
$(\text{Confirm (Yes/No)})_{HTTPs}$
Step 11
https://127.0.0.1:4116/evoting/thankyoupage.html
Step 12
Encrypts and stores the vote:
$(\text{Vote})_{PUBVA}$
$(\text{Vote})_{PUBV}$
Step 13
Requests HTTPs connection establishment by remote server
Step 14
$((\text{Vote}))_{PUBVA})_{HTTPs}$
Step 15
SMS, IVR call, email

Figure 4.3: SCWS Voting Protocol

Table 4.3: OWASP Vulnerabilities and Defences

OWASP Top Ten Reference	**Is it relevant to SCWS?**	**Defences**
A1 Injection	Yes	Traditional injection attacks such SQL or LDAP (Lightweight Directory Access Protocol) do not apply here. However injection attacks targeting the login page where the input fields are located can occur. Strict filtering of input to this page will be needed.
A2 Broken Authentication and Session Management	Yes	All credentials are encrypted and stored inside the SIM card where it is very difficult for them to be retrieved. Additionally, all credentials go through secure TLS connections while in transit and there is no account creation/password changing directly on the SIM. If an attacker were to gain possession of the credentials, they are useless anywhere except on the phone where they are stored. So the attacker has to be in possession of the phone or manage to access the SCWS remotely (although according to the specification the SCWS is not designed for such remote access)
A3 Cross-site Scripting (XSS)	Yes	The only input fields that the attacker can use are in the login page. Strict filtering of input to this page will be needed.
A4 Insecure Direct Object References	Yes	Assuming that the applications on the SCWS may have different users accessing the same application or different levels of access (admin vs regular/standard user), this attack definitely applies and proper authorisation and verification of all access requests should exist.
A5 Security Misconfiguration	No	This attack does not apply to the SIM environment, as it is expected that the SCWS will be properly configured and patched by the MNO. In general, this threat exists when systems that work with a web server (databases, operating systems, etc.) may be left unpatched and insecure.

OWASP Top Ten Reference	**Is it relevant to SCWS?**	**Defences**
A6 Sensitive Data Exposure	No	It is expected that the tamper resistance of the SIM card and the use of secure communication channels, will protect data in transit and in rest. All sensitive data (votes) will be encrypted, as an added precaution in case storage space limitations give rise to a requirement to store them in a non-tamper resistant memory location, or on the phone. Any access to the cryptographic keys could be protected by a PIN (or not given at all), and the tamper resistance capabilities of the SIM should resist physical attacks. All communication will occur over HTTPS (VA to agent, browser to SCWS) and all certificates are valid.
A7 Missing Function Level Access Control	Yes	Considering the fact that the SCWS will host applications that offer different levels of access, proper authorisation functions must exist in each.
A8 Cross-Site Request Forgery (CSRF)	Yes	This attack possibly applies to the SCWS environment, as a malicious user can use a CSRF attack to access a session that the legitimate user may already have with the SCWS.
A9 Using Known Vulnerable Components	No	As with A5 it is expected that all SCWS components will be properly patched and secure. Another factor that provides mitigation against this type of attack, is that it will be very difficult for a potential attacker to identify such vulnerabilities, as the SCWS will not by publicly accessible.
A10 Unvalidated Redirects and Forwards	Yes	This attack has the potential of being launched against the SCWS, e.g. if an application forwards the user to another webpage, when this webpage is or can be provided or manipulated by the user (which also means that an attacker can use this attack to redirect the user to some malicious website) by providing a GET parameter that is used by the web application to identify the redirect location. Safe redirects, as per web development best practices, must be used to mitigate such attacks.

in PAV). Finally, a flag should be set to indicate that the voter has voted.

The remote administration server passes the signed vote (unchanged) to the VA, tagged with a temporary ID so that the vote receiving process at the VA does not find out the mobile number, as this could be used to link the voter to a particular vote. Once the VA has received the vote, checked its validity, and posted it to the bulletin board, it will recalculate the confirmation code for the chosen candidate, and send it and the temporary ID to the remote administration server. The confirmation code can then be returned to the voter via SMS, for checking against the one stored on the SCWS. This provides assurance that their vote was counted as cast because the correct confirmation code can only be generated at the VA once it has been successfully decrypted and matched to a valid entry in the bulletin board. A dummy vote process would obtain all the confirmation codes for all the candidates via SMS: the dummy ballot form would then be discarded, and a new one selected in its place.

The verification procedures of PAV gives the voter the assurance that their vote has been cast as intended, and counted as cast. Using the SCWS model keeps all sensitive vote information private in a tamper resistant environment, so that no one can determine how a voter voted.

The next section now provides a preliminary analysis of the security of the proposal.

4.5 Security Analysis

4.5.1 Web Server Security

To analyse the security of the e-voting solution, it is necessary to consider the main characteristics of the overall approach and how it deals with attacks and risks, especially those which target web applications. The Open Web Application Security Project (OWASP) [88] issues a document which includes the Top Ten vulnerabilities that affect the security of web applications, and as the SCWS will serve web content to clients on the phone, it faces the same challenges and considerations as traditional web servers. However, most of these identified vulnerabilities will not affect the security of the proposed solution because the attack surface of the SCWS is small, and the code/data used by the application is held inside a secure token. Table 4.3 shows the Top Ten vulnerabilities, whether they are relevant to the security of the voting site stored on the SCWS and how the proposal defends against them.
In contrast, traditional remote e-voting schemes have faced many of the common internet application problems such as DoS attacks [74] and SQL injections [73]. Additionally, 'secure platform' [77] issues arise because the voter's equipment cannot be trusted to perform as expected either through malware or system vulnerabilities.

Table 4.4: E-Voting Attacks and Defences

Attack Goal	**Method**	**Defences**
Find out contents of vote, or how a voter voted	Retrieve vote from SCWS	Two copies of the vote are stored inside the SIM card, one encrypted using PUB_V and the other by PUB_{VA} (this is deleted once the vote is cast). So the vote is always held encrypted in the secure token and inaccessible to an attacker. The SCWS can only be accessed from the handset's HTTP client and the remote administration server, thereby minimizing the likelihood of attacks that occur remotely. An attacker would need to know the voter's credentials in order to access the SCWS voting site.
Change contents of vote	Tamper with data in transit	The vote in transit is protected by the security of HTTPS for all communication channels.
	Tampering by the browser	By using the PAV system, any type of tampering with the vote can be detected by the use of confirmation codes and a Bulletin Board where the voter may check their vote was included as cast.
Prevent voter from voting	Denial of Service attack	Distributing the voting application to the SCWS makes the system resilient to centralised web server attacks.
	Steal the phone	There will need to be procedures in place to allow poll-site voting for those whose handsets are lost/stolen/malfunctioning.
	Suppress the vote: do not send it to VA from the SCWS	This attack is very difficult to launch successfully, as it needs to target individual phone handsets. Using an SMS message to send a confirmation code back to the voter will give assurance that their vote has indeed been retrieved and sent to the voting authority. If no SMS is received then the voter can raise a query with the VA, and use an alternative method to vote.

Attack Goal	Method	Defences
Vote more than once	Create fake voters IDs	This will be handled by the registration procedure and is out of scope of this chapter.
	Vote on several phones	A voter's credentials (voter ID/password) are installed directly onto the SCWS via the MNO Full Administration Protocol. The credentials can only be used to access one particular SCWS: if an attacker found out a voter ID/password, e.g. through coercion, they would not work with a different SCWS as they would not be recognised. Also, the VA must have back office procedures to make sure that a voter ID is not used to vote more than once.
	Vote multiple times on one phone	If an attacker obtains multiple voter ID/passwords, they cannot be used to vote on a single handset, as the application on the SCWS will only recognise the registered voter's credentials. If the attacker has gained access to the registered voter's ID/password during the voting process, the credentials could not be used to vote on the same phone again, as the ballot form will only be displayed when a voter ID has not been used before.

4.5.2 Smart Card Web Server Security

The main security characteristics of using a SCWS based voting solution are as follows:

1. There is no centralised voting server to be attacked, hence a single point of failure has been removed. DDoS attacks cannot easily take place because the voting procedure is dispersed among many voters. Each voter effectively has a voting server on their SIM, so for an attack to have a significant outcome, the attacker has to infiltrate a large number of phones and find a way to orchestrate an attack against them. A potential attacker must also find out which phones are registered for voting in order to target them.

2. The SCWS remote administration server could be considered a single point of failure, but the important fact here is that this server may have restricted functions and access to it is controlled. It runs on the Trusted Third Party (TTP)/MNO premises, so should be isolated from any unauthorised physical access. However, although internal threats cannot be ignored, within the scope of this proposal it is assumed that the MNO is trusted to provide both the infrastructure and management procedures securely.

3. The SCWS, the voting site and the credentials will be stored in a secure token, the SIM card. The SIM has tamper resistant capabilities which can offer defences to mitigate against physical and side channel attacks. Even if an attacker manages to circumvent the considerable defences that the SIM presents they will only gain access to one voter's credentials. The voting application code will be stored in the SIM, thus the attacker will need advanced knowledge and significant effort to gain access to it. Again, the attacker would have to target a large number of phones and their SCWS/SIMs to be effective.

4. The entire communication takes place over standard secure communication channels, protected by the HTTPS protocol. As mentioned in point 3, the voter's credentials and the rest of the important components of the scheme are securely stored in a tamper resistant token. Thus the integrity of the entire scheme can be reasonably assured.

4.5.3 E-Voting Security Issues

Voting systems must ensure that votes are: cast as the voter intended; recorded as cast; counted as recorded; and not linkable to a specific vote. Also, no one should be able to determine how a voter voted. An attacker can have many goals, including manipulating the votes randomly, denying access to the voting procedure for legitimate voters, adding votes for a specific candidate/party (ballot stuffing), spoiling votes for a particular candidate or gaining knowledge about a voter's choice of candidate. By its nature, remote voting is vulnerable to coercion and vote buying, where a voter votes (willingly or unwillingly) as instructed by a third party. Coercion is a generic problem which is not solved by the proposal in this chapter: other remote e-voting approaches may be better suited to high-coercion scenarios. Table 4.4 shows a summary which demonstrates how the proposed system deals with the most relevant attacks against its security requirements.

There are many areas which could benefit from further research, some of which are outlined in Chapter 8.

We believe that our solution can be used for additional purposes as well and e-voting is only one use case that will help us explain the whole concept and feasibility.

4.5.4 Our Solution Against DDoS Attacks

The distributed nature of our proposal can be a new approach for the protection against DDoS attacks that get bigger and bigger. Mitigating these attacks can be very expensive, but in some scenarios it may not even be feasible. This protection can be applied to cases like banking or other critical applications where downtime is impossible to tolerate. People must be able to conduct banking even when the site is offline. The claim that a bank cannot go offline is not valid as it is very difficult to maintain 100 per cent security or availability. Specifically for the use case presented in this thesis, there have been instances of attacks against voting websites, e.g. against the Canadian New Democratic Party Elections in 2012, while the 2016 US presidential elections were full of articles of potential DDoS attacks during the elections. These attacks did not materialise, but DDoS attacks are now part of the electoral discussions. Our approach removes the necessity for a single point of failure. A voting website will not be served from a specific location only, but rather the voting procedure will be distributed across a large number of mini-sites.

4.6 Scyther Tool

The protocol described in this chapter was analysed by a verification tool called Scyther [89]. Scyther is a tool used for the verification, falsification, and analysis of security protocols. It is an open-source tool and it is very fast compared to other solutions. More information on the Scyther tool can be found in [90] and information on the verification algorithm in [91]. Scyther takes a protocol as input and evaluates its 'security claims', i.e. its security properties (e.g. a secret value) and outputs information on certain attacks against the protocol and its claims. Scyther was used in this thesis (as opposed to other protocols) based on the fact that it is faster than most of the other protocols, that its installation is very easy compared to e.g. Casper and it has a smaller learning curve. There are also examples available and in general the tool is quite well documented and fairly easy to use, without compromising on the correctness of the results that it provides. A comparison of the efficiency of the various verification tools can be found in [92]. It must be noted that Scyther was used to find vulnerabilities in the ISO/IEC 9798-2 Entity Authentication standard [93]. This resulted in an updated version of the protocol. Scyther was also used to identify weaknesses and to suggest improvements on ISO/IEC 11770 [94]. Finally, all examples in this thesis were peer reviewed for correctness.

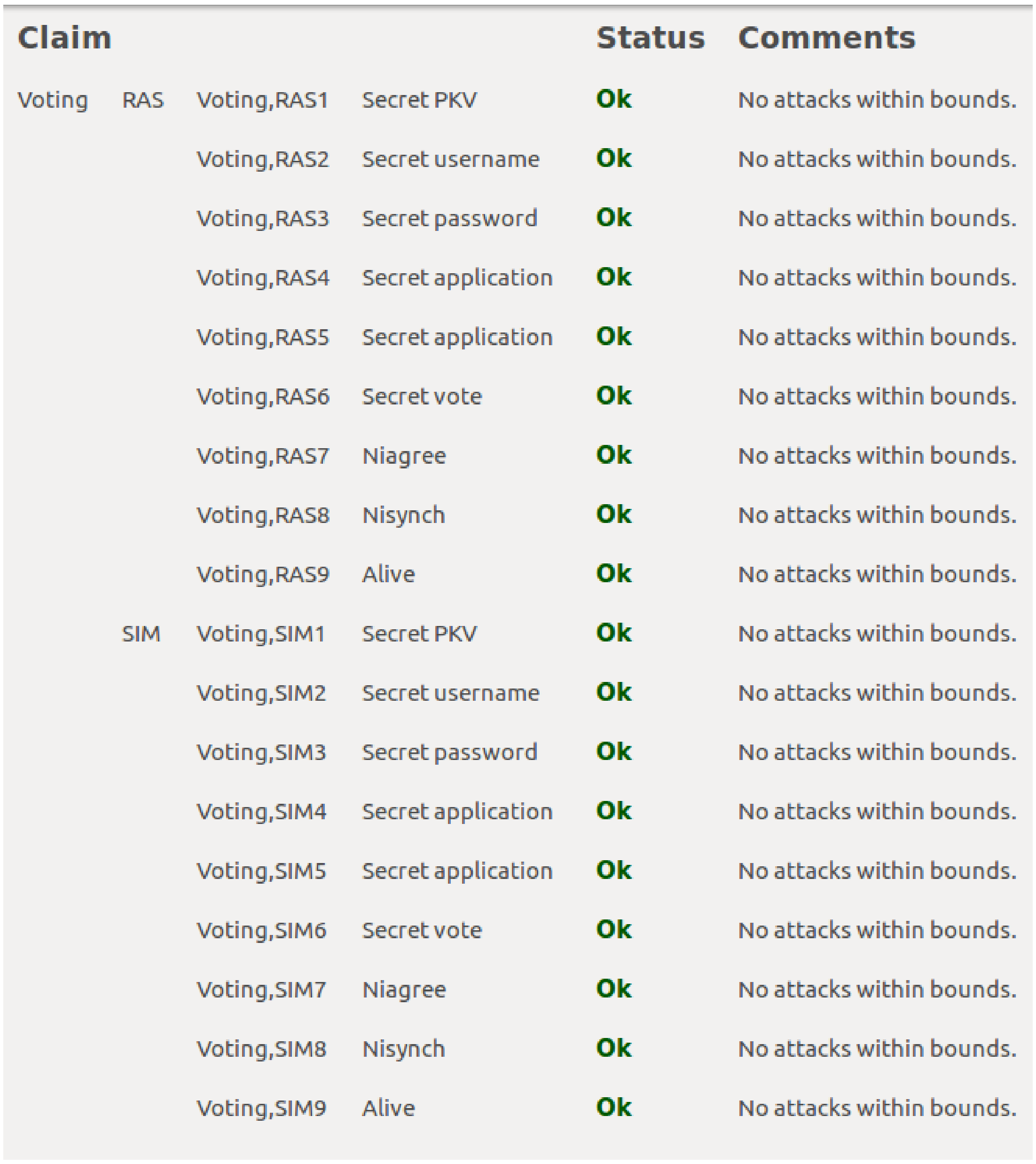
4.7 Formal Analysis of the Protocol

In Figure 4.4 we present the results of the formal/verification analysis of the protocol discussed in this chapter - the code can be found in Appendix A. The figure describes the basic entities that participate in the protocol (RAS and SIM card, i.e. SCWS), presents their claims and if there are attacks against any of these claims. If there was an attack found, it would have been mentioned here and the user would be able to see the attack by clicking on a button. As shown in the figure, the Scyther tool did not find any attacks on our protocol as shown by status 'OK' (either values being secret or messages being exchanged).

4.8 Conclusion of the Chapter

This chapter has presented a method for using a phone equipped with a Smart Card Web Server SIM as a secure voter interface for e-voting, using the Prêt à Voter scheme as an example. Distributing web server functionality to voters' SIM cards means that there is no central web server to target, and so an attacker must compromise many phones to successfully affect the election result. The use of the SIM's tamper resistant environment for the storage and processing of sensitive voter credentials also addresses the secure platform problem.

In the next chapter, we will present the idea of using the SCWS to secure card-not-present transactions on the Web. The SCWS and the SIM card can be used during the checkout process of an e-commerce transaction. The benefit is for the user to not having to submit any cardholder data to the merchant and for the merchant to not having to store any of these data, thus minimising the PCI DSS compliance requirements.



Claim				Status	Comments
Voting	RAS	Voting,RAS1	Secret PKV	Ok	No attacks within bounds.
		Voting,RAS2	Secret username	Ok	No attacks within bounds.
		Voting,RAS3	Secret password	Ok	No attacks within bounds.
		Voting,RAS4	Secret application	Ok	No attacks within bounds.
		Voting,RAS5	Secret application	Ok	No attacks within bounds.
		Voting,RAS6	Secret vote	Ok	No attacks within bounds.
		Voting,RAS7	Niagree	Ok	No attacks within bounds.
		Voting,RAS8	Nisynch	Ok	No attacks within bounds.
		Voting,RAS9	Alive	Ok	No attacks within bounds.
	SIM	Voting,SIM1	Secret PKV	Ok	No attacks within bounds.
		Voting,SIM2	Secret username	Ok	No attacks within bounds.
		Voting,SIM3	Secret password	Ok	No attacks within bounds.
		Voting,SIM4	Secret application	Ok	No attacks within bounds.
		Voting,SIM5	Secret application	Ok	No attacks within bounds.
		Voting,SIM6	Secret vote	Ok	No attacks within bounds.
		Voting,SIM7	Niagree	Ok	No attacks within bounds.
		Voting,SIM8	Nisynch	Ok	No attacks within bounds.
		Voting,SIM9	Alive	Ok	No attacks within bounds.

Figure 4.4: E-Voting Protocol Formal Analysis (1/1)

Chapter 5

Card-Present Transactions on the Internet Using the Smart Card Web Server

Contents

This chapter is based on the paper *Card-Present transactions on the Internet Using the Smart Card Web Server*, 12th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2013), Melbourne, Australia.

In the previous chapter we presented a novel way to allow the distribution of a service (e-voting), in order to avoid DDoS attacks and to disperse the execution of such application, across multiple phones. We took advantage of the secure execution environment of the SIM card that is able to protect all necessary secrets required for the conduct of an important process like e-voting. In this chapter we will see, how we can use the same approach to secure payments and transactions on the Web. The idea is to allow the user to proceed with a payment in a card-not-present environment, such as on the

Web, by actively requesting her to be in possession of a SIM card that contains the card number and is able to participate in the transaction or to have the actual card. This is achieved by the use of redirections between the e-commerce site and the SCWS that is found inside the SIM card. Eventually, the card-not-present transactions are transformed into card-present ones.

5.1 Introduction

In many countries the Internet is considered a necessity and access is almost ubiquitous [95] [96]. People enjoy access to the rich content and services that the Internet offers, when at work, at home and increasingly when on the move through the use of smartphones and other portable devices like tablets. Purchasing over the Internet is also increasing in popularity [97], both on traditional computing and on mobile platforms. Merchants typically operate e-commerce websites adopted for the large screens of PCs, although increasingly they are adapting to the small smartphone devices. According to the survey by Forrester Research, the estimated transaction volume is to grow to €172 billion by 2016 from €96.6 billion in 2011 [97]. However, this growth can be proven fragile since a large number of security concerns exist - this fact may undermine the confidence of customers.

Unfortunately, the exchange of large amounts of money and the billions of transactions every year, attract malicious groups and individuals that aim to steal valuable customer data and indeed their money. These targets include credit card data that are then used directly for fraudulent purchases or sold on to other criminals. Malicious entities target both users and merchants using malware, such as trojan horses and keyloggers [98]. A common practice for merchants is to store cardholder data in their back end systems for various reasons (recurring transactions, dispute, etc.). However, this practice has vulnerabilities and in the past there were incidents of massive data exposure due to attacks targeting merchants [99]. In some instances data leaked not as a result of an attack, but due to operational mistakes [100]. This led the five major card brands (VISA, MasterCard, American Express, JCB, Discover) to create the Payment Card Industry Data Security Standard (PCI DSS) [60] aimed at raising merchant security. Merchants are required to be certified and the ones that do not comply to the standard may face large fines, increased charges and/or the removal of the facility to accept payments by credit/debit cards [101]. Additionally, on the 'user side' there are commercial programs that claim to improve security of sensitive data submission on the Internet like e.g. Kaspersky's 'Safe Money' [102] or Trusteer's Rapport [103]. However and despite these attempts, credit/debit card fraud is still a large problem for e-commerce [104].

The proposal described in this chapter achieves its aims through the use of a credit/debit card (for traditional PCs) or SIM card (for phones) that embeds a SCWS.

The basic advantages of this proposal are:

1. Users that wish to shop online, either from a phone or a PC, are not required to submit any card details. They are just expected to know a PIN number and be in possession of the card.
2. Merchants never receive any cardholder data, instead they receive a single use virtual credit card number (for an example see [105]). As a result, merchants do not have to store or protect any card data in their back end databases/systems, thus they are less affected by PCI DSS requirements [106]. The economic advantage of avoiding a number of PCI DSS requirements depends on the size of the company, but it is certainly an important factor.
3. The cardholder data are transferred encrypted at all times and are only visible to the issuing bank.
4. There is no need for third parties e.g. PayPal.
5. Only standard protocols are used (HTTPS) [107].
6. There is no need for installation of any special application for the user, since only a standard browser is required.
7. Card data are stored inside a widespread and proven tamper resistant security token; a smart card.
8. The use of the Remote Administration Server (in the case of the SIM card) allows for the secure transfer of data, information, code and anything else required for the solution to work (the entire communication takes place over HTTPS).
9. New possibilities are opened for the convenience of the user e.g. a user can browse the SCWS and check previous payments/transactions.

The chapter is structured as follows: Section 5.2 outlines the current payment solutions for PCs and smartphones, Section 5.3 presents the proposed architecture, Section 5.4 is a security analysis of the proposal, while Section 5.5 discusses possible attacks and how they are mitigated by this proposal. Section 5.6 provides a formal analysis of the protocol and Section 5.7 concludes the chapter.

5.2 Other Solutions

A lot of related work, (both academically and commercially driven) is taking place regarding e/m-shopping/payments security. We differentiate between e-commerce and m-commerce, with the former using a traditional computing device like a PC or a laptop, and the latter using mobile devices (mainly smartphones and tablets). Note that a smartphone and its browser can be used for conventional e-commerce, whereas a desktop PC typically cannot support m-commerce.

5.2.1 Electronic Shopping

A typical sequence of actions during an electronic shopping (e-shopping) session is:

1. The user checks out after choosing the goods/services she wants to purchase.
2. Then the user is prompted to select a payment method. Most common choices include paying by a credit/debit card or using a payment service e.g. PayPal.
3. Once the user authorises the payment, the card details are send to the issuing bank, through the acquirer and the payment brand. The issuing bank verifies the ability of the customer to pay or not and the transaction concludes.

The payment methods are crucial to transaction security and to the disclosure of customer data, and so the options are now discussed in more detail.

1. A merchant may allow the customer to submit card details (the card number i.e. Primary Account Number, cardholder's name, card expiration date, Card Verification Value(CVV)/Card Verification Code (CVC)) directly to the merchant's website. The merchant then submits to the bank the details for transaction authorisation without storing them.
2. There are websites on the Internet that allow a user to save the details of a normal credit/debit card, to simplify future purchases, as then the user is just asked to login to the website and confirm the payment. A common example of such a website is Amazon [108].
3. Using a payment service such as PayPal or Google Wallet [109] [110]. The merchant supports this type of payment through the inclusion of a button that redirects the customer to the payment service's website. In order for a transaction to proceed, the user has to have an account with the service (including a pre-registered payment card). The security of the whole procedure depends on the password that the shopper creates and uses to login to the service. Once logged in, she is not required to submit any card details (they are stored in the service's databases); she just confirms the payment.

4. Using prepaid cards (e.g. PayPal's prepaid card [109]). The shopper has to top-up the card credit before using it to purchase goods. The advantage of using cards of this type is that even if the card details are stolen, the person loses only the limited amount of card credit. The payment procedure is the same as with credit/debit cards, i.e. the user has to submit the card and delivery details. It must be noted however that there are merchants/shops on the Internet that do not accept payments with prepaid cards and that some payment services and/or banks stopped offering this service.

5. Using disposable cards (called virtual credit cards or one-time cards). The virtual/disposable card is linked to the actual credit card and the user has to use a client program (most commonly a browser) to contact the issuer in order for the latter to create and send her the one-time credit card details. The card is for a single use only and cannot be used by an attacker after the transaction concludes. Our proposal uses a single use virtual credit card [105].

6. Due to implications and mandates of PCI DSS and instances of data breaches, a number of merchants are no longer willing to store payment data. They prefer to use the services of a payment gateway, which acts on their behalf, accepting the card details from the user, submitting it for authorisation and informing the merchant. The user is asked to submit the card details either to a new website that pops up or to an inline frame within the merchant's website. This frame is operated by the payment gateway and the merchant does not receive any card details [111].

5.2.2 Mobile Shopping

Mobile shopping (m-shopping) offers newer payment methods for customers. Using a phone, a person is able to pay in-shop using Near Field Communication (NFC) [112] [113], which for example allows the phone to emulate payment cards or to read external cards. Alternatively, the phone can be used to make m-payments to merchants via the Internet, which is the focus for the research described in this chapter. A person is able to pay using her phone in a variety of ways, including the following examples.

1. Using SMS messages. In most instances the customer is required to send one (or more) SMS messages to premium numbers in order for her to be able to pay for the product/service. In some instances a user receives a PIN number after sending the required SMS and inputs this PIN as proof of payment [114].

2. Through direct mobile billing. Instead of paying using a card, a user may choose to include the purchase amount on her next mobile phone bill. The user effectively defers payment to the time when she pays for her mobile phone bill [115].

3. Using e-payment solutions. This means that a user browses normally through a website and when required to pay, she uses one of the conventional solutions mentioned above (in the e-payments section).
4. Using special applications that run on mobile platforms (Android, iOS, etc.). An example is Amazon applications that can be used with most of the popular mobile operating systems.

The payment solutions described thus far do not make use of the SCWS which is a fundamental component of our solution. In the next section, we will describe in detail the architecture of the proposed solution.

5.3 Proposed Architecture

The proposal presented in this chapter can be implemented on two different platforms, both of which must be able to host a SCWS:

1. On a SIM card.
2. On a 'normal' smart card like a credit or debit card.

The SCWS described in the OMA specification is used in conjunction with a SIM card, thus the SIM card acts as its host and the SCWS can be used with a phone. To our knowledge there is no mention anywhere for a Smart Card Web Server that will run inside a normal credit/debit card. A SIM card web server (as we can call an SCWS of the first category) does not need anything else apart from the SIM and the phone to operate, while for the credit/debit card web server (an SCWS of the second category) a reader or a relevant slot (for smart cards) on the PC/laptop is needed.

Another major difference between the two choices is that in the case of the credit/debit card solution the key management and the distribution of the cards/cardholder data is relatively simple since the only entities involved are the user and the issuing bank (the issuing bank or 'Issuer' is the bank that originally issues the credit/debit card to the user and is responsible for checking if a particular credit/debit card can be used for a certain transaction - the 'Acquirer' is the bank that has a contractual agreement with the merchant and deposits the transaction money to its account after the payment is authorised by the issuing bank). This is not the case for the SIM based approach, since for a user to receive the cardholder data in her SIM card, there have to be agreements and explicit trust between the bank and the MNO. The MNO is unavoidably enmeshed in the whole procedure since in some way it will be involved, either by providing space in the SIM card for the issuing bank to fully administer or just being a 'mere conduit' for the transfer of the card data on the SIM. In this case the MNO acts as the issuer of the ‘hardware’ i.e. the SIM card and the bank as the issuer of the data that will be transferred and stored on the SIM by the MNO. For the purposes of this chapter and whenever the term ‘Issuer’ is used we mean the issuing bank.

The proposed protocol flow remains the same for both options, although option 1 (on a SIM card) is currently being tested and SIM cards of this type already exist experimentally, whereas option 2 (on a ‘normal’ smart card) is considered to be a new proposal of this chapter.

5.3.1 Basic Assumptions

We claim that the protocol offers the desired level of security, while at the same time maintains its usability and ease of use for the common user. However, some basic assumptions have to be made prior to the description of the protocol itself.

1. For the security of data-in-transit we rely on HTTPS. While HTTPS is the de facto solution for the Internet, any severe vulnerabilities found will affect the security of our solution. On the same level, support for HTTPS from all entities involved is unquestionable, although for the communication between the acquirer, payment brand and issuer other protocols or architectures (private networks, virtual or not) may be used. In any case data-in-transit should be protected to the best possible level.

2. For this proposal to work, we rely on a pre-existing hierarchical PKI infrastructure. The issuing bank will act as the signed certificate provider for the cards, while the merchant will obtain a certificate signed by its acquirer. Both the issuer and the acquirer will have their certificates signed by a common and trusted Payment Brand such as Visa or Mastercard. An issuer would check a merchant certificate signature, by first verifying the signature on the acquirer certificate. This way the issuer will have reasonable assurance that is communicating with a valid merchant and not some bogus entity. It is expected that all certificates follow the X.509 standard [116].

3. The last part of the message exchange (between the acquirer and the issuer through the Card Payment Network) relies on existing architectures and is generally considered to be secure. In general both the acquirer and the issuer trust the payment brand and the latter is the connecting and trusted point between the entities.

4. Settlement and arrangement of payment are out of scope for this chapter, however common practices are expected to take place.

5. Finally, when the SCWS is stored on the SIM card, we have to consider the MNO as a trusted entity (in the sense that it will not undermine by any means

the security of the data stored on the card) and assume that all the necessary countermeasures and contractual agreements are in place. In such occasion the MNO will be considered as a 'mere conduit'.

Table 5.1: E-Payments Notation

Notation	**Description**
USR	User
MER	Merchant
ISS	Issuer/Issuing Bank
ACQ	Acquirer/Acquiring Bank
SCWS	Smart Card Web Server
MID	Merchand ID
MNAME	Merchant Name
TDLS	Transaction Details (amount, date, time, merchant's transaction ID)
TREF	Issuer's Transaction Reference
PAN	Primary Account Number
TPAN	Temporary PAN
EXP_DATE	Card Expiration Date
CNAME	Cardholder Name
CVV/CVC	Card Verification Value (3 or 4 digits number)
CNTR	Freshness Counter
AUTH_REQ	Authorisation Request
AUTH_RES	Authorisation Result
T_RES	Transaction Result
PUB_{Iss}	Issuer Public Key
PUB_{Mer}	Merchant Public Key
PUB_{Acq}	Acquirer Public Key
PR_{Iss}	Issuer Private Key
PR_{Mer}	Merchant Private Key
PR_{Card}	Card Private Key
Cert_{Iss}	Issuer Certificate
Cert_{Mer}	Merchant Certificate
Cert_{Acq}	Acquirer Certificate
SIG	Signature of Data Using a Private Key

5.3.2 Description of the Protocol Flow

Below follows a description of the protocol flow depicted in Figures 5.1 and 5.2. Additionally the reader can find all relevant notations in Table 5.1.

1. A user browses the merchant's website and goes to checkout. She chooses to pay

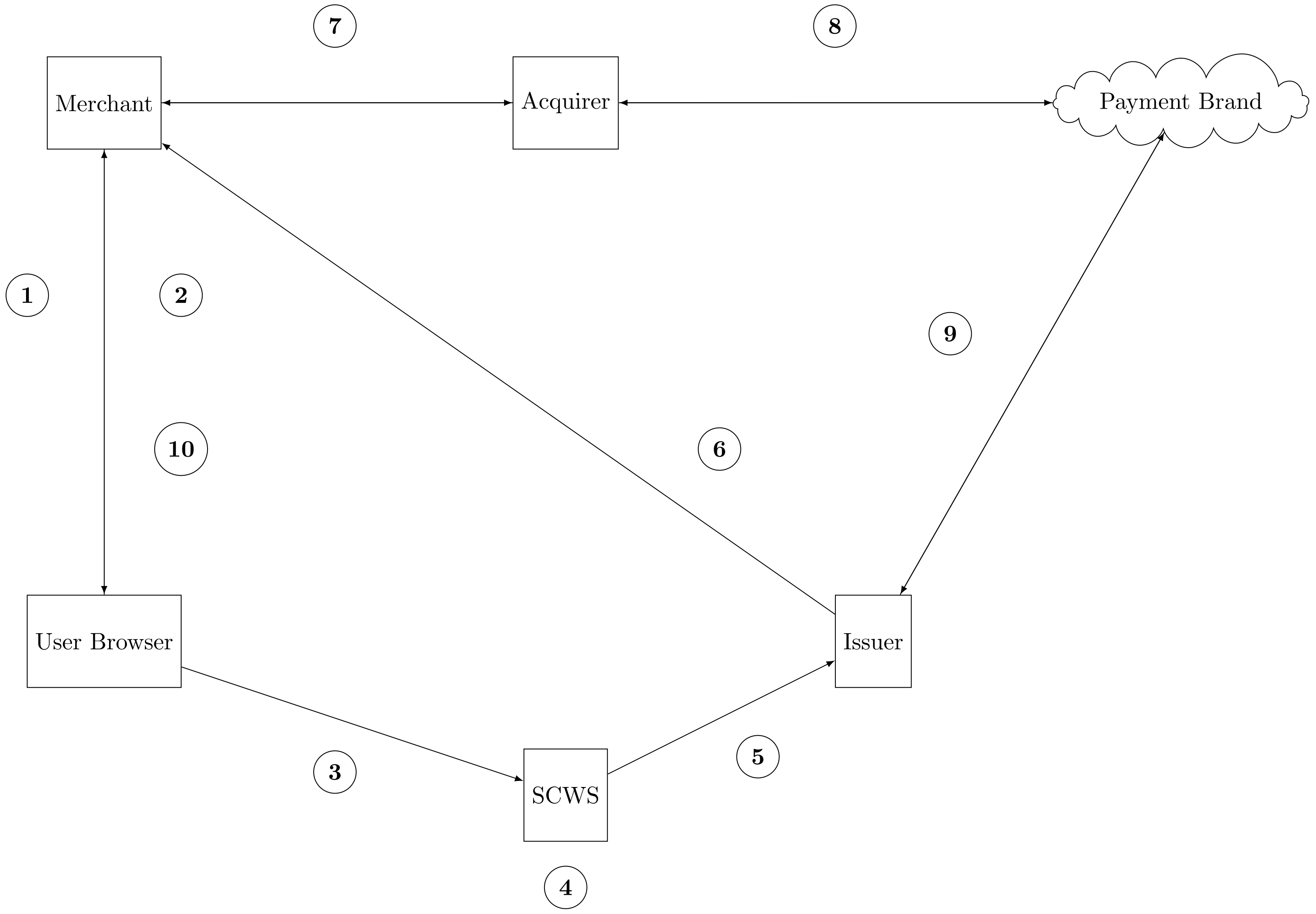


Figure 5.1: E-Payments Protocol Diagram

using the SCWS and informs the merchant about this by selecting the appropriate choice.

2. The merchant prepares the link that points to the SCWS and which includes all the necessary information for the transaction i.e. the amount, date/time, a Transaction ID (we call these the Transaction Details - TDLS) and two URLs that the bank will later use (step 6) to redirect the user's browser in case of successful or unsuccessful result. The merchant signs this information using its private key ($\mathrm{PR}_{\mathrm{Mer}}$) and attaches its certificate $\mathrm{Cert}_{\mathrm{Mer}}$ and the acquirer's certificate $\mathrm{Cert}_{\mathrm{Acq}}$ (that is used to verify the merchant's certificate). We name this string of data as MTOKEN. The merchant presents the link to the user through her browser. The latter clicks the link and is transferred to the SCWS environment. The entire communication takes place over HTTPS. It has to be mentioned that the browser is just the standard PC or phone browser.

3. The user is now in the SCWS environment (meaning that the web page presented to her is created by the SCWS) and is asked for her PIN. She inputs the PIN in her browser and this is transmitted to the SCWS over HTTPS.

4. If the PIN is correct, a Java servlet running inside the card, retrieves the card data that reside in the card and prepares a token that includes: the Primary Account Number (PAN) of the card, the Expiration Date (EXP_DATE), the CVV/CVC, the Cardholder's Name (CNAME) and an updated value of a counter (CNTR) that will help mitigate replay attacks. It then creates a signature of all the above information using the card's Private Key($\mathrm{PR}_{\mathrm{Card}}$) and encrypts everything using the issuer's public key ($\mathrm{PUB}_{\mathrm{Iss}}$). We name this cryptogram as STOKEN. The SCWS presents a link that points to a pre-stored URL/Domain Name of the issuing bank.

5. The user is presented with the link and once she clicks it, she is transferred to the issuing bank's domain.

6. The issuing bank receives the information, i.e the MTOKEN and the STOKEN. It goes through the necessary decryption and verification process, thus retrieving the card data and the TDLS and verifying the attached signatures. It does all the necessary back end checks to ensure whether the card can be used for the transaction (for example if the card can pay for the amount): if so, it creates a single use virtual card number (TPAN) that is associated with both the original card number (PAN) and this specific transaction. It also adds a transaction reference for future acknowledgement (TREF). This data is signed using the issuer's private key ($\mathrm{PR}_{\mathrm{Iss}}$) and encrypted using the merchant's public key ($\mathrm{PUB}_{\mathrm{Mer}}$) - the issuer also attaches its public key certificate ($\mathrm{Cert}_{\mathrm{Iss}}$). Finally, it redirects the user to the MID_URL_YES. If for any reason the transaction is declined, the user

is redirected to the MID_URL_NO and neither TPAN nor TREF is created and sent.

7. The merchant receives this cryptogram (in case of a positive reply from the issuing bank) and decrypts it using its private key (PR_{Mer}), thus receiving the TPAN and TREF - it also checks the validity of the signature using the issuing bank's public key (retrieved from the certificate). Otherwise and in case of a negative reply from the issuing bank, it informs the user and either prompts her to use another payment option, try again or simply rejects the entire transaction and terminates the session. If the merchant proceeds with the transaction, it sends the necessary information to the acquirer. The message that is sent includes the MID, MName, the TPAN, the TREF and any other information necessary for the acquirer to proceed. It then creates a signature of all the above information using its private key(PR_{Mer}) and encrypts everything using the acquirer's public key (PUB_{Acq}).

8. The acquirer receives the cryptogram and decrypts it. If the necessary checks e.g. on the MID, MNAME, signature created by the merchant, are correct and the signature is valid, it sends an Authorisation Request (AUTH_REQ) to the issuer. To do so, it forwards the transaction to the payment brand's network (e.g. VISANet).

9. The issuer receives the info from the payment brand and checks the pair TPAN and TREF against the ones that were originally created (step 6). If the check is successful, the issuer informs the payment brand and through it the acquirer and the merchant (Authorisation Result - AUTH_RES).

10. The merchant displays the transaction result to the user (Transaction Result - T_RES) and completes the process.

The next section provides a security analysis of the aforementioned architecture.

5.4 Security Analysis

The security of this proposal is outlined in this section:

- The biggest difference for a person that uses this method for payment (compared to traditional ways), is that she never has to submit her cardholder data during a transaction. The Primary Account Number (PAN), the Cardholder's Name, the CVV/CVC and the Expiration Date never leave the environment of the smart card in unencrypted form.

- All data in transit is protected by HTTPS. Although HTTPS suffers from certain vulnerabilities [117], it is still the de facto solution for secure transfer of data on

Step	**Source**	**Message**	**Destination**
1	USR	$\xrightarrow{Checkout}$	MER
2	MER	$\xrightarrow{(TDLS \Vert MID\_URL\_YES \Vert MID\_URL\_NO \Vert (SIG)_{\mathrm{PR_{Mer}}} \Vert Cert_{\mathrm{Mer}} \Vert Cert_{\mathrm{Acq}})_{\mathrm{HTTPS}}}$	USR
3	USR	$\xrightarrow{(PIN)_{\mathrm{HTTPS}}}$	SCWS
4	SCWS	$(PAN \Vert EXP\_DATE \Vert CVV \Vert CNAME \Vert CNTR \Vert (SIG)_{\mathrm{PR_{Card}}})_{\mathrm{PUB_{Iss}}}$	
5	SCWS	$\xrightarrow{(MToken \Vert SToken)_{\mathrm{HTTPS}}}$	ISS
6	ISS	$\xrightarrow{((TPAN \Vert TREF \Vert (SIG)_{\mathrm{PR_{Iss}}})_{\mathrm{PUB_{Mer}}} \Vert Cert_{\mathrm{Iss}})_{\mathrm{HTTPS}}}$	MER
7	MER	$\xrightarrow{(MID \Vert MNAME \Vert TPAN \Vert TREF \Vert (SIG)_{\mathrm{PR_{Mer}}})_{\mathrm{PUB_{Acq}}}}$	ACQ
8	ACQ	$\xrightarrow{AUTH\_REQ}$	ISS
9	ISS	$\xrightarrow{AUTH\_RES}$	ACQ/MER
10	MER	$\xrightarrow{T\_RES}$	USR

Figure 5.2: E-Payments Protocol Flow

the Internet. Almost every website that requires a certain level of security (e-banking, e-commerce/business, authentication, etc.), relies on the security offered by HTTPS.

- In case that HTTPS is proven to not provide the necessary level of security (if new, severe vulnerabilities are found), the cardholder data is always transmitted encrypted using algorithms compliant with information security best practices [118].

- Replay attacks are defended against by the use of a freshness counter.
- Alteration of data is difficult to achieve since messages are encrypted and signed and even if successful can be detected.
- Only the issuer sees the cardholder data: it is not available to the merchant or any other party. Thus, if a user pays exclusively by this method for the entire life of the card (until its expiration date) the cardholder data will never be submitted anywhere other than back to the issuer (and always encrypted).
- User convenience is enhanced without compromising the security of the proposal. Currently, a user should submit card details every time she needs to purchase something or risk reliance on a third party (e.g. PayPal). This proposal requires the user to simply click on two browser links and submit a PIN number, thus achieving a practical balance between user convenience and security.
- The implications for merchants are positive too. In conventional transactions, a user is typically required to submit the cardholder data to the merchant, for the latter to proceed with the transaction. However, this makes the merchants responsible for the safekeeping of the data. Unfortunately there have been numerous instances in the past, where cardholder data was successfully retrieved by attackers or simply lost, to the penalty of merchants and customers alike. Under the new proposal, a merchant would not have to store or process any raw card data, thus avoiding the burden associated with PCI DSS compliance [13].
- Cardholder data is protected against attack and disclosure while stored within the SCWS as we benefit from the physical security and tamper resistance of the smart card. Even if smart cards can be attacked, a malicious entity should be in possession of the card itself.
- PIN discovery is always a possibility as we rely on the mobile device and keyloggers, trojan horses, general malware, remote attacks (and even phishing) might develop to capture and export the user PIN, however this would be of little use to attackers unless they also have possession of the associated smart card.

5.5 Possible Attacks/Mitigation

Just like any other payment solution, this proposal has to be as secure as possible to mitigate any possible attempted attacks against its security and ultimately, that of the cardholder data. Some types of attack and potential mitigation are discussed below.

1. *Modification:* This attack is difficult to achieve, since data in transit is signed.

2. *Eavesdropping:* This should not be possible, since data transfers are encrypted using algorithms and keys compliant with best practices [118].

3. *DoS attacks:* Malware could theoretically attack the SCWS in order for it not be able to receive/process/send the data. However for this to have a widespread effect the attacker would have to target many user phones (if the SCWS is running in the SIM card).

4. *Attacks against the PIN:* Even if malware is able to steal the PIN number, this is practically useless since the attacker has to be in possession of the card to perform transactions.

5. *Attacks against the card:* Smart cards are widespread and proven security tokens. Although they are not tamper proof (attacks are reported against them e.g. [119]), they can be extremely tamper resistant, rendering most forms of attack practically infeasible.

6. *Replay attacks:* These attacks [120] are difficult to launch since the *STOKEN* includes a freshness counter. Even if malware prevents the browser from sending this token and steals it, the outcome is not in favour of the attacker (she cannot retrieve, alter or use the encrypted data).

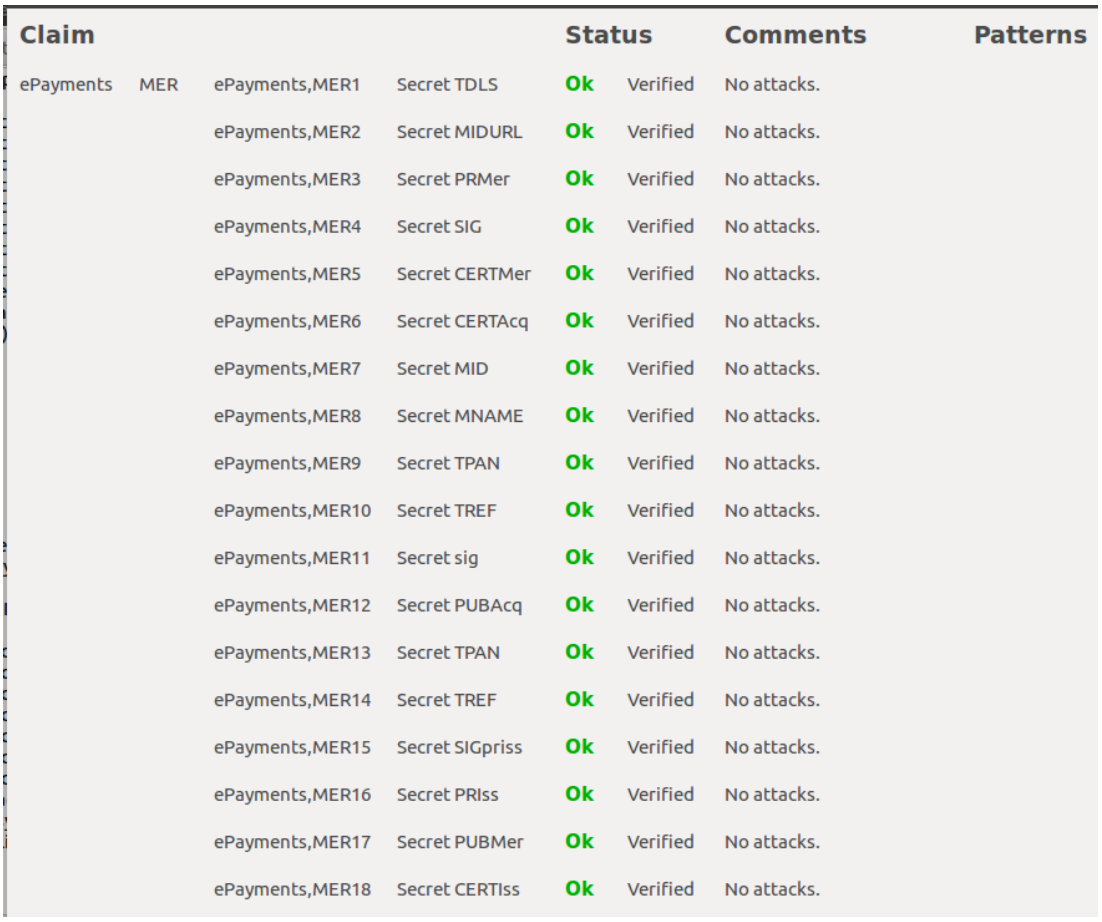
7. *Phishing websites*: A user may still be tricked to visit a bogus website however this will not result in any of the SCWS stored data being disclosed, removing any benefit for the attacker.

8. *Merchant insider attacks*: Such attacks are less of a concern (compared to a conventional approach) as the merchant no longer holds cardholder data.

5.6 Formal Analysis of the Protocol

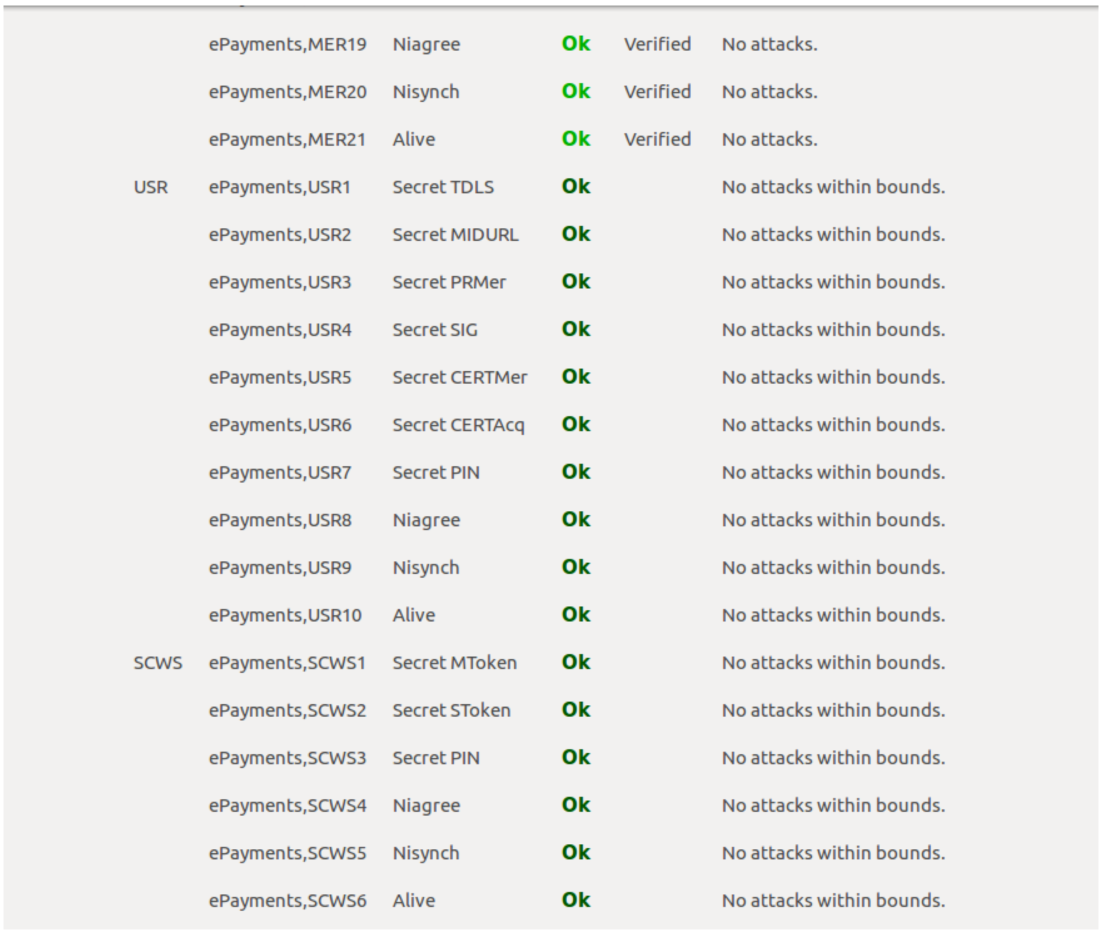
For the formal/verification analysis of the protocol discussed in this chapter, we used the code presented in Appendix B. The basic entities are MER (merchant), USR (user), SCWS, ISS (issuer) and ACQ (acquirer). Next to them are the claims for each of these entities and possible attacks are defined in the last two columns. See below the analysis of the protocol in Figures 5.3, 5.4, 5.5 and 5.6. Scyther also identified an attack as shown in Figure 5.7.

This attack affects step 7 of the protocol and assumes that the attacker sends fake data to the acquirer i.e. TREF, TPAN, MName and MID, signs them with her private key and then encrypts everything with a fake public key. We believe that the latter shows that the tool did not properly identify what is happening in this step. Encrypting something with the attackers public key does not offer anything to the attacker.



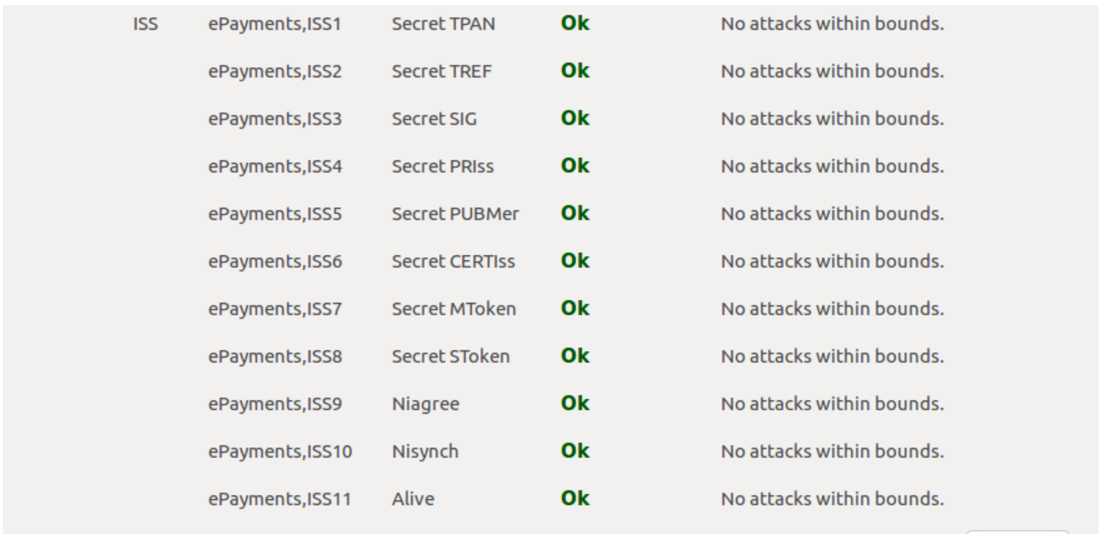
Claim				Status		Comments	Patterns
ePayments	MER	ePayments,MER1	Secret TDLS	Ok	Verified	No attacks.	
		ePayments,MER2	Secret MIDURL	Ok	Verified	No attacks.	
		ePayments,MER3	Secret PRMer	Ok	Verified	No attacks.	
		ePayments,MER4	Secret SIG	Ok	Verified	No attacks.	
		ePayments,MER5	Secret CERTMer	Ok	Verified	No attacks.	
		ePayments,MER6	Secret CERTAcq	Ok	Verified	No attacks.	
		ePayments,MER7	Secret MID	Ok	Verified	No attacks.	
		ePayments,MER8	Secret MNAME	Ok	Verified	No attacks.	
		ePayments,MER9	Secret TPAN	Ok	Verified	No attacks.	
		ePayments,MER10	Secret TREF	Ok	Verified	No attacks.	
		ePayments,MER11	Secret sig	Ok	Verified	No attacks.	
		ePayments,MER12	Secret PUBAcq	Ok	Verified	No attacks.	
		ePayments,MER13	Secret TPAN	Ok	Verified	No attacks.	
		ePayments,MER14	Secret TREF	Ok	Verified	No attacks.	
		ePayments,MER15	Secret SIGpriss	Ok	Verified	No attacks.	
		ePayments,MER16	Secret PRIss	Ok	Verified	No attacks.	
		ePayments,MER17	Secret PUBMer	Ok	Verified	No attacks.	
		ePayments,MER18	Secret CERTIss	Ok	Verified	No attacks.	

Figure 5.3: E-Payments Protocol Formal Analysis (1/4)



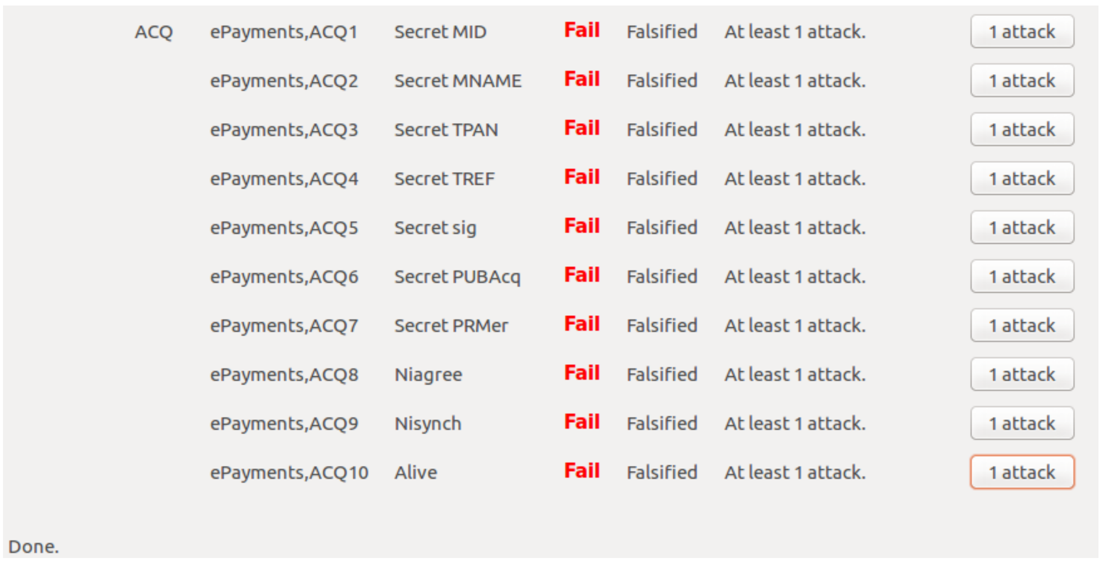
	ePayments,MER19	Niagree	Ok	Verified	No attacks.
	ePayments,MER20	Nisynch	Ok	Verified	No attacks.
	ePayments,MER21	Alive	Ok	Verified	No attacks.
USR	ePayments,USR1	Secret TDLS	Ok		No attacks within bounds.
	ePayments,USR2	Secret MIDURL	Ok		No attacks within bounds.
	ePayments,USR3	Secret PRMer	Ok		No attacks within bounds.
	ePayments,USR4	Secret SIG	Ok		No attacks within bounds.
	ePayments,USR5	Secret CERTMer	Ok		No attacks within bounds.
	ePayments,USR6	Secret CERTAcq	Ok		No attacks within bounds.
	ePayments,USR7	Secret PIN	Ok		No attacks within bounds.
	ePayments,USR8	Niagree	Ok		No attacks within bounds.
	ePayments,USR9	Nisynch	Ok		No attacks within bounds.
	ePayments,USR10	Alive	Ok		No attacks within bounds.
SCWS	ePayments,SCWS1	Secret MToken	Ok		No attacks within bounds.
	ePayments,SCWS2	Secret SToken	Ok		No attacks within bounds.
	ePayments,SCWS3	Secret PIN	Ok		No attacks within bounds.
	ePayments,SCWS4	Niagree	Ok		No attacks within bounds.
	ePayments,SCWS5	Nisynch	Ok		No attacks within bounds.
	ePayments,SCWS6	Alive	Ok		No attacks within bounds.

Figure 5.4: E-Payments Protocol Formal Analysis (2/4)



ISS	ePayments,ISS1	Secret TPAN	Ok	No attacks within bounds.
	ePayments,ISS2	Secret TREF	Ok	No attacks within bounds.
	ePayments,ISS3	Secret SIG	Ok	No attacks within bounds.
	ePayments,ISS4	Secret PRIss	Ok	No attacks within bounds.
	ePayments,ISS5	Secret PUBMer	Ok	No attacks within bounds.
	ePayments,ISS6	Secret CERTIss	Ok	No attacks within bounds.
	ePayments,ISS7	Secret MToken	Ok	No attacks within bounds.
	ePayments,ISS8	Secret SToken	Ok	No attacks within bounds.
	ePayments,ISS9	Niagree	Ok	No attacks within bounds.
	ePayments,ISS10	Nisynch	Ok	No attacks within bounds.
	ePayments,ISS11	Alive	Ok	No attacks within bounds.

Figure 5.5: E-Payments Protocol Formal Analysis (3/4)



ACQ	ePayments,ACQ1	Secret MID	Fail	Falsified	At least 1 attack.	1 attack
	ePayments,ACQ2	Secret MNAME	Fail	Falsified	At least 1 attack.	1 attack
	ePayments,ACQ3	Secret TPAN	Fail	Falsified	At least 1 attack.	1 attack
	ePayments,ACQ4	Secret TREF	Fail	Falsified	At least 1 attack.	1 attack
	ePayments,ACQ5	Secret sig	Fail	Falsified	At least 1 attack.	1 attack
	ePayments,ACQ6	Secret PUBAcq	Fail	Falsified	At least 1 attack.	1 attack
	ePayments,ACQ7	Secret PRMer	Fail	Falsified	At least 1 attack.	1 attack
	ePayments,ACQ8	Niagree	Fail	Falsified	At least 1 attack.	1 attack
	ePayments,ACQ9	Nisynch	Fail	Falsified	At least 1 attack.	1 attack
	ePayments,ACQ10	Alive	Fail	Falsified	At least 1 attack.	1 attack

Done.

Figure 5.6: E-Payments Protocol Formal Analysis (4/4)

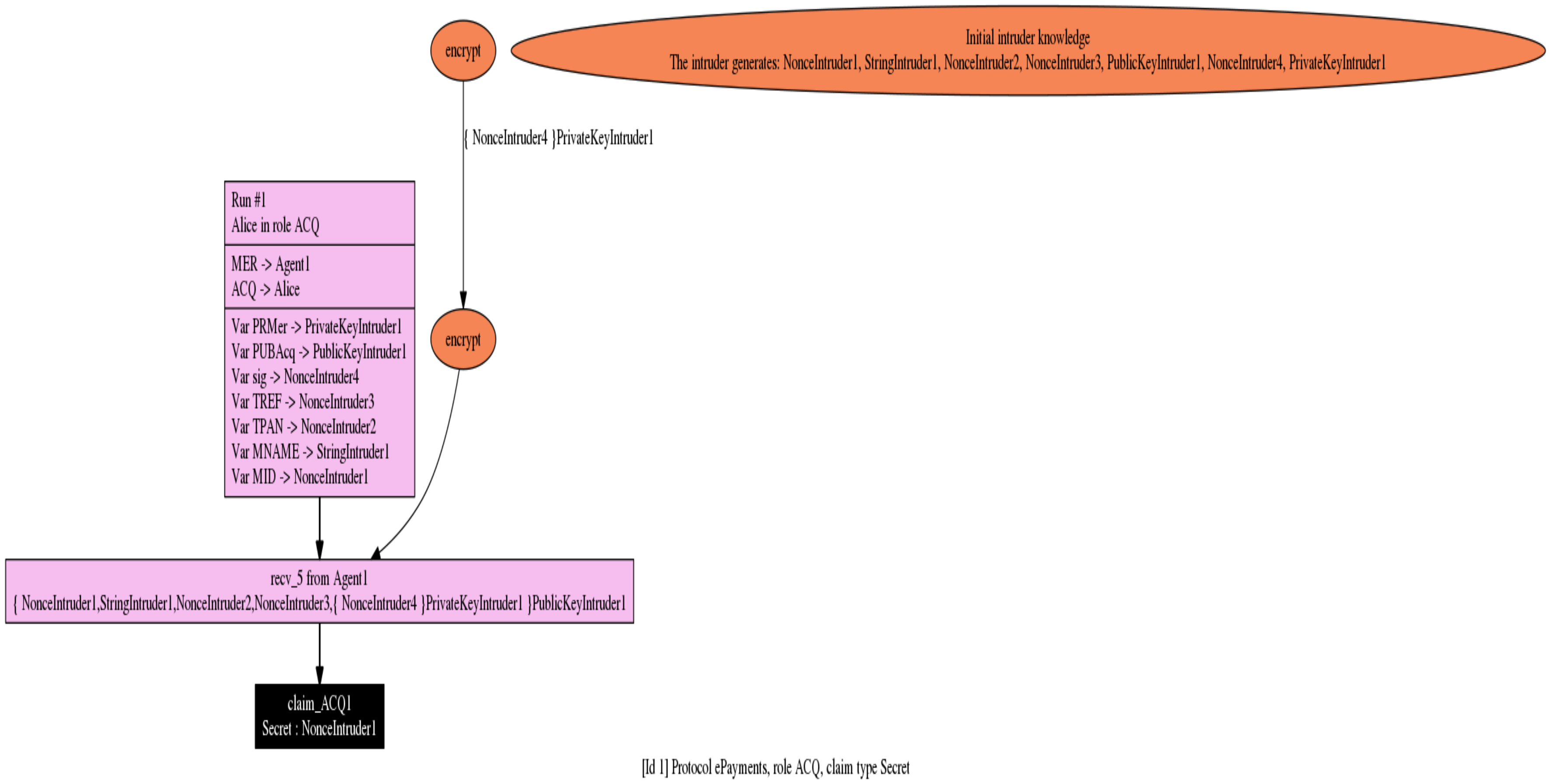

encrypt
Initial intruder knowledge
The intruder generates: NonceIntruder1, StringIntruder1, NonceIntruder2, NonceIntruder3, PublicKeyIntruder1, NonceIntruder4, PrivateKeyIntruder1
{ NonceIntruder4 }PrivateKeyIntruder1
Run #1
Alice in role ACQ
MER -> Agent1
ACQ -> Alice
Var PRMer -> PrivateKeyIntruder1
Var PUBAcq -> PublicKeyIntruder1
Var sig -> NonceIntruder4
Var TREF -> NonceIntruder3
Var TPAN -> NonceIntruder2
Var MNAME -> StringIntruder1
Var MID -> NonceIntruder1
encrypt
recv_5 from Agent1
{ NonceIntruder1,StringIntruder1,NonceIntruder2,NonceIntruder3,{ NonceIntruder4 }PrivateKeyIntruder1 }PublicKeyIntruder1
claim_ACQ1
Secret : NonceIntruder1
[Id 1] Protocol ePayments, role ACQ, claim type Secret

Figure 5.7: E-Payments Protocol Attack

The conclusion of this chapter is now described in the section that follows.

5.7 Conclusion of the Chapter

In this chapter we saw a way of using the SCWS to secure payments and transactions on the Web. The user does not need to submit any cardholder details anywhere, a fact that raises the security of payments on the Internet.

In the next chapter we will expand the idea of using the SCWS in a way to support and secure the authentication process for a user of a virtual world. The use of a SIM card and of an SCWS, will allow for an easy integration with a virtual world environment, without compromising the security of the entire process.

Chapter 6

Virtual World Authentication Using the Smart Card Web Server

Contents

This chapter is based on the paper *Virtual World Authentication Using the Smart Card Web Server*, International Symposium in Security in Computing and Communications (SSCC'13), Mysore, India.

In the previous chapter we presented a way to allow payments on the Web by using the SIM or the debit/credit card as a means to conduct transactions in a card-present way. The user does not need to submit or remember any cardholder data or provide these data when trying to purchase on the Web, thus raising the protection level for both cardholders and merchants alike. This chapter will depict how we can use the SCWS

Haltestelle der Strassenbahn
Alexanderplatz Unter den Linden
83
TEPPICHHAUS Lefevre Oranienstr. 15A

Figure 6.1: Virtual World[1]

during the authentication process on the Web. A use case that will depict such functionality is when the user wants to authenticate against a virtual world environment.

6.1 Introduction

The term 'Virtual World' (VW) has been defined as 'a synchronous, persistent network of people, represented as avatars, facilitated by networked computers' [121]. VWs are very popular: at the beginning of 2012, there were 1,921 million registered accounts in over 100 VWs [122]. Some VWs aim to mimic real life as closely as possible in a digital environment e.g. Second Life, by Linden Labs [123]. Others are designed as game environments, where the objective is to complete quests and enhance your avatar's skills and reputation, for example Blizzard's World of Warcraft [124]. Figure 6.1 shows a screenshot of a VW.

Authentication procedures for VW users are currently fairly limited, mostly relying on static, easily compromised username/password combinations. This can result in a number of security issues with real world consequences e.g. virtual goods/identities can be stolen if an account is hacked, with the danger that the user's computer can then be compromised or real world identity theft could occur [125].

Using a mobile phone as a second factor in online authentication (i.e. something you have/something you know) should improve security. There are several existing mobile phone authentication options e.g. using SMS messages [127], or optical challenge-

[1]Image by Jo Yardley, distributed under a Creative Commons, Attribution-Share Alike 3.0 Unported license [126]. No modifications made to the image.

response procedures [128]. Mobile phone operating systems are increasingly becoming targets for malware, however [129]. Storing credentials in a tamper resistant device, such as the SIM card, is therefore a more attractive approach. This chapter proposes a system that uses a SIM equipped with a SCWS in a one-time password (OTP) authentication scheme. Including geolocation [130] in the protocol introduces another obstacle for a potential attacker to overcome, by checking if the phone and PC are in the same geographical area. The aim is to significantly improve the security of the VW login, without unduly inconveniencing the user, and so enhance user and merchant confidence in VW security and services offered to users.

The chapter is structured as follows: the technical architecture of VWs is outlined in Section 6.2 and the proposed protocol is described in Section 6.3. A preliminary security analysis of the proposal is done in Section 6.4, a formal analysis of the protocol using the Scyther tool is presented in Section 6.5, before the chapter reaches its conclusion in Section 6.6.

6.2 Virtual World Infrastructure

To connect to a VW the user needs client software, typically installed on a PC, represented as C1-C5 in Figure 6.2. The VW Client (VWC) can be regarded as a special kind of viewer or browser, in that the client gets data from a remote server, and renders it into a visual representation for the user to interact with. The VW environment is provided by the VW infrastructure i.e. a VW cloud consisting of a set of servers. There are load balancers, front end servers (denoted as S1, S2, S3 in Figure 6.2), and back end facilities usually composed of databases/VW rules which apply to interactions with objects/avatars inside the world. This infrastructure is subdivided into components and processes, used to conduct particular tasks and introduce modularity [131] [132]: e.g. a login component, a user component and a data component [133]. The proposal in this chapter is concerned with the login server processes which handle the initial request from the user to log into the world.

Now that VW technical architecture has been described, the proposed protocol will be detailed in the next section: however, the necessary entities and assumptions, along with OTP processing and geolocation techniques will be outlined first.

6.3 The Proposed SCWS Authentication Protocol

6.3.1 Entities and Assumptions

Entities: The entities in the proposal are described in Table 6.1, and their relationship is shown in Figure 6.3. Notation used is in Table 6.2.

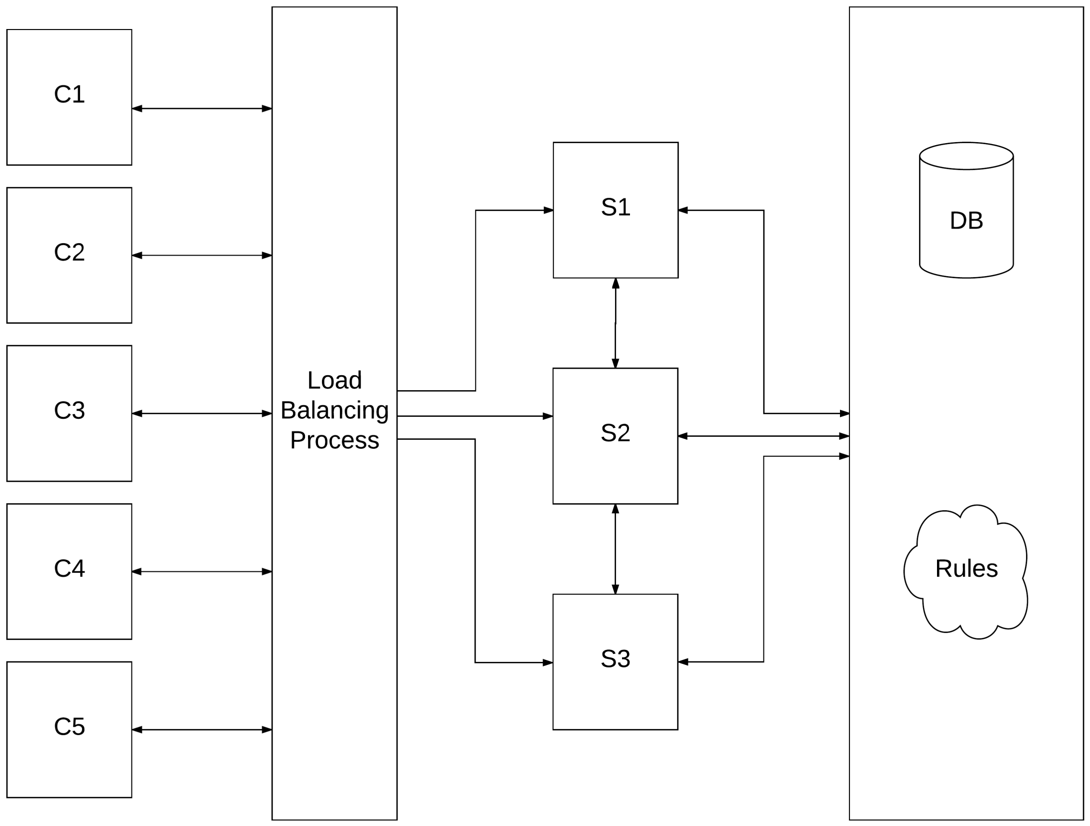


Figure 6.2: Virtual World Infrastructure

Assumptions: The following assumptions are made:

1. ***Business relationships:*** The MNO has a business relationship with the VW developers, and has authorised their use of a Remote Administration Server to update the SCWS VW application and data.

2. ***Users per SCWS:*** The mobile phone has a one-to-one mapping to the user, i.e. only one registered user can use a particular SCWS.

3. ***Registration:*** A secure user registration procedure is in place. The exact details are out of the scope of this chapter, but a user must supply credentials such as mobile phone number/PIN so that the RAS can download the application/-credentials onto the correct phone and the PIN number can be used to create the OTP during the protocol. The VW authentication Java servlet will then be installed on the user's SIM, along with the user's PIN to access the SCWS. Additionally, during registration, a VW certificate will be installed on the user's VW Client to offer mutual authentication.

4. ***Channel Security:*** Channels between the RAS/SCWS, and VWC/VWS are considered secure, since all data in transit is transferred over HTTPS.

Table 6.1: Description of Entities

Notation	Description
VW	The virtual world. Has details of all user credentials. It is responsible for checking user login details and creating one-time passwords.
VWS	Virtual World Server. Provides back end functionality for the VW, with all necessary information to keep the VW operating. It connects with back end processes such as the database and the login server.
LS	Login Server. Part of the virtual world, this manages login details, authenticates users, creates the nonce (N) and OTP, then checks the OTP sent by the VWC.
SCWS	Smart Card Web Server. The user accesses the SCWS environment using a PIN. The SCWS uses a Java servlet to process and display the OTP.
AA	Administration Agent: an on-SIM entity that establishes a HTTPS connection with the RAS.
VWC	Virtual World Client. This is the interface presented to the user, installed on the user's PC. It provides just a graphical user interface, and is considered insecure.
RAS	Remote Administration Server. It updates the SCWS of a registered phone via the MNO's Full Administration Protocol, using HTTPS. It is a trusted entity, and can be operated by the MNO or a trusted third party. The RAS can also determine the phone's location (as it is a part of the MNO).
U	User. An individual who is registered with a VW.

6.3.2 One-Time Passwords and Geolocation

One-Time Passwords: Many VWs use a simple username/password as the user authentication mechanism at login, relying on static data for security; this may be captured and exploited via a range of IT security attacks. The general principle behind the proposed protocol is that an OTP generating process is done both at the VW back end and by the SCWS, using a nonce N and the user's PIN. Thus the use of a static password is replaced by that of a dynamically created one on a separate personal device, using a tamper resistant chip (the SIM), and connected via a different communications channel. Having an OTP generator on the VW server side means it benefits from the VW server's existing security, which is reasonably expected to be of a considerably higher level than on the VW client. The OTP should have a number of security properties: e.g. easy to compute, but very difficult to identify the PIN/ nonce used to create it; have a minimum length of 64 bits; and be created using cryptographically secure, standardised pseudo-random number generators such as RFC4226 [134]. A compromise of usability and security will be needed: e.g. a user may be prepared

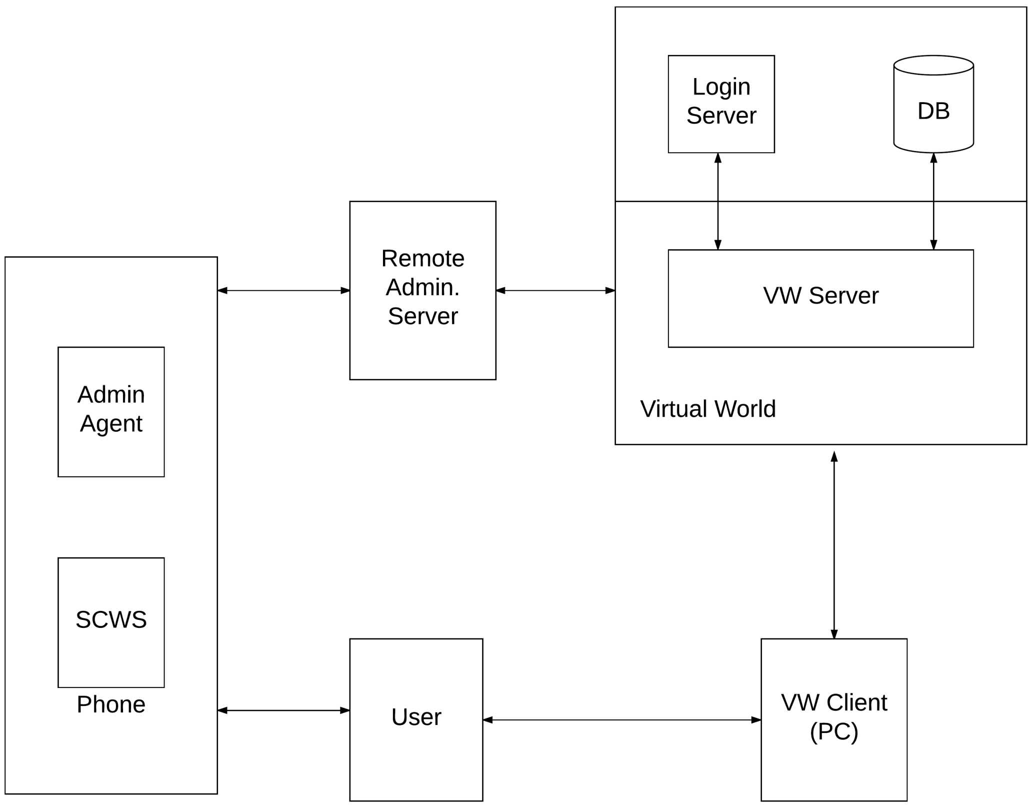


Figure 6.3: Relationship Between Entities

to type in a maximum of eight alphanumeric digits and so the OTP must be mapped to this field size; PIN size is crucial for the entropy of the OTP, however a small PIN size increases usability, but decreases security (and vice versa). Common practice for financial institutions and MNOs is to use sizes of 4 and 5 digits, with the maximum number of PIN entry attempts limited to 3 to protect against exhaustive (brute force) attacks .

Geolocation: The phone and VWC's locations are also used as a further obstacle for a potential attacker. The phone's location could be determined by either the cell towers of the mobile network or through the GPS adapter. Equivalently IP geolocation could be used to find the client's location. The authentication is successful if both locations are within a certain distance of each other. While IP geolocation is not extremely accurate, it is currently used by large organisations to counteract attacks originating some distance from the genuine user.
Note: privacy issues arising from the use of geolocation services are out of scope for this thesis.

6.3.3 The Authentication Protocol

The steps in the protocol are now described (see Figures 6.4 and 6.5).

Table 6.2: Virtual Worlds Authentication Notation

Notation	**Description**
F	Function used for one-time password generation
N	Nonce
ID	Any form of user identity, like a username or other identifier
PIN	The PIN number that the user uses to login to the SCWS
OTP	One-Time Password: The OTP that is generated as the result of function F in both the login server and the SCWS
Loc_{VWC}	The location of the VWC
$\text{Loc}_{\text{Phone}}$	The location of the phone

Step 1: The user initiates the VWC installed on their PC, and is requested to input their unique ID credentials (e.g. VW username/user ID).
Step 2: The VWC software initiates a secure connection (HTTPS) with the VWS. Both the VWS and the VWC have a digital certificate, so that mutual authentication is achieved. The client software sends the ID credentials to the VWS over this secure channel.
Step 3: The VWS identifies this as a secure login request and passes it to the LS.
Step 4: The LS checks the ID credentials: e.g. whether the user is already logged in, or has been banned due to some previous illicit action. If the credentials are acceptable, a time constrained authentication protocol is initialised. The LS creates a nonce (N), and an OTP in combination with the user's PIN i.e. *OTP = f(N,PIN).* This OTP is stored securely in the LS. The VWS has substantial power, so OTP generation will not impact its performance. It must be noted that the OTP is only valid for a limited time period.
Step 5: The LS transmits N back to the VWS. Since both these servers are internal to the VW they are assumed to be secure and trusted.
Step 6: The VWS forwards N to the RAS. These entities also trust each other, and the connection between them is also established over a secure channel.
Step 7: The RAS determines the location of the phone ($\text{Loc}_{\text{Phone}}$) and returns it to the VWS.
Step 8: The VWS forwards the $\text{Loc}_{\text{Phone}}$ to the LS.
Step 9: Once it receives N from the VWS, the RAS will establish an HTTPS connection with the user's SIM card and transfer N to the SCWS.
Step 10: Once N has been securely transferred to the SCWS, the user will connect to the SCWS via a link shown on the phone's browser. The user will be asked to input their PIN: if correct the OTP generation process is initialised.
Step 11: The PIN number is used as input to the same function used in the LS along with the nonce N that was received during step 9, to reproduce the OTP. This number is transformed into an eight byte alphanumeric string.

Step 12: The SCWS returns this string to the phone's browser via HTTPS.
Step 13: The user inputs the displayed OTP manually into the VWC. This is a time-constrained action; the LS must receive the OTP within a certain period.
Step 14: The entered OTP will be forwarded to the VWS (along with VWC's location Loc_{VWC}) over the previously established HTTPS connection.
Step 15: When the OTP/Loc_{VWC} pair reaches the VWS and is verified as authentic, it is forwarded to the LS.
Step 16: When the LS receives OTP/Loc_{VWC} pair, it will compare the OTP against the previously created OTP (step 4) and check the location of the PC client Loc_{VWC} against the location of the phone $\text{Loc}_{\text{Phone}}$ from (step 8)
Step 17: Successful checks authenticate the user on the LS: the VWS is notified.
Step 18: A message is sent to the VWC saying that the user is logged in. The VWC refreshes to the standard layout and user enters the VW.

A preliminary security analysis of the protocol will now be conducted.

6.4 Security Analysis of the Protocol

The objective of this protocol is to give an assurance that if avatar X is in the virtual world then we can reasonably assume that its legitimate real world controller (user X) has logged in by using two-factor authentication. The protocol also takes into consideration usability as one of the factors of security. There are several potential attack points which need to be considered with regard to the protocol's overall security, and these will be now analysed.

6.4.1 SCWS and Mobile Phone

Physical security of the SIM: The SCWS is only accessible from the RAS or the phone browser and a malicious party needs to have physical access to the mobile phone to retrieve security information sent from the RAS. The information is processed by an servlet on the tamper resistant SIM/SCWS. To retrieve information about the servlet/algorithms/nonces used, etc, an attacker has to physically attack a SIM card or to know the correct SCWS PIN.

Web based attacks against the SCWS: The SCWS is a web server and as such the OWASP Top Ten attacks [88] still apply (as shown in Table 4.3), but the potential to mount these from the phone browser is limited and remote access to the SCWS is only permitted from the trusted RAS. The small attack surface of the restricted SCWS environment means that attackers have fewer opportunities to attack compared to a 'traditional' web server. All web pages stored on the SCWS have little processing or scripting in them, which minimises attack potential. For more information, please see

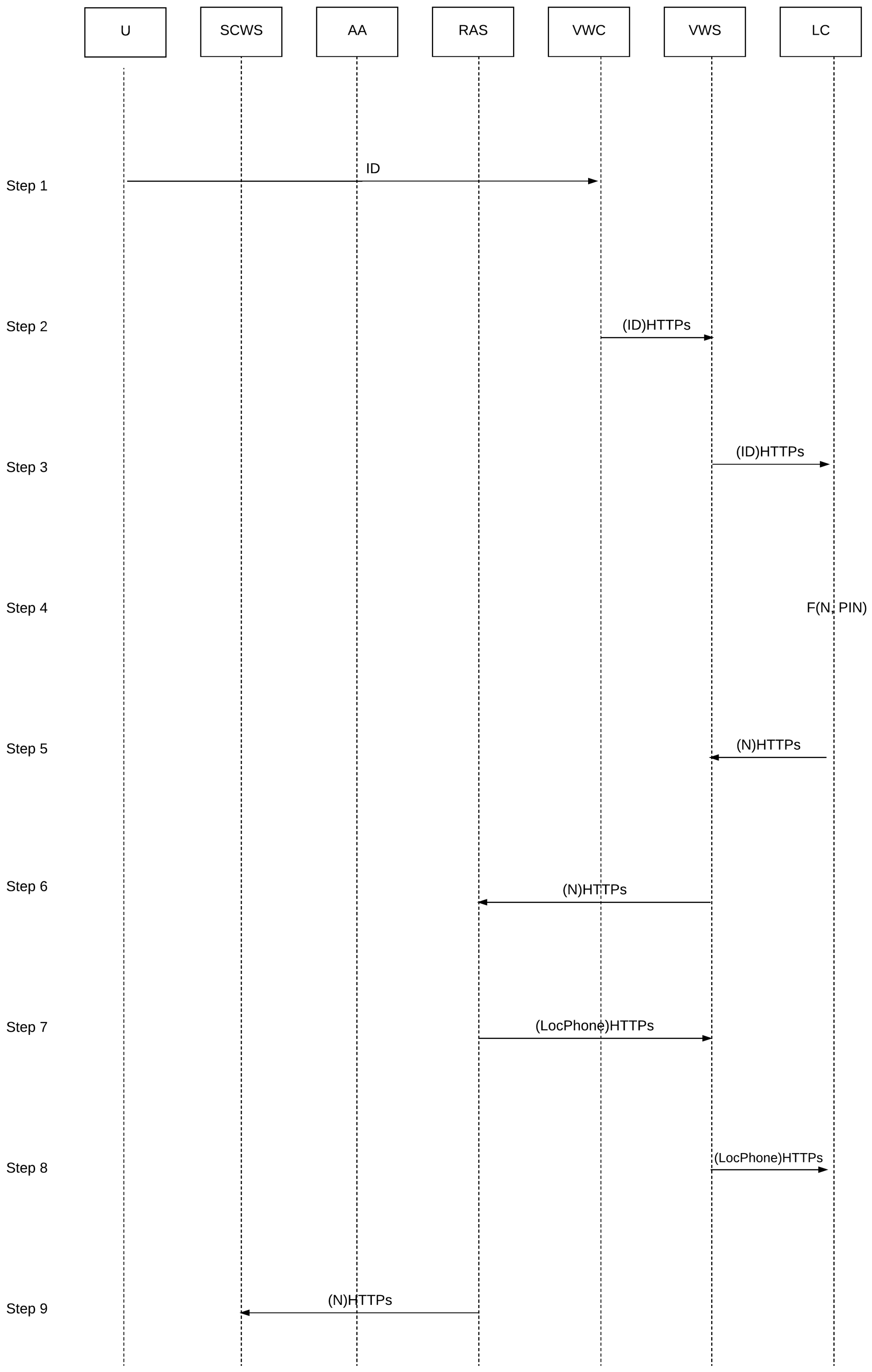


Figure 6.4: Virtual World Protocol Flow (Steps 1-9)

Table 4.3.

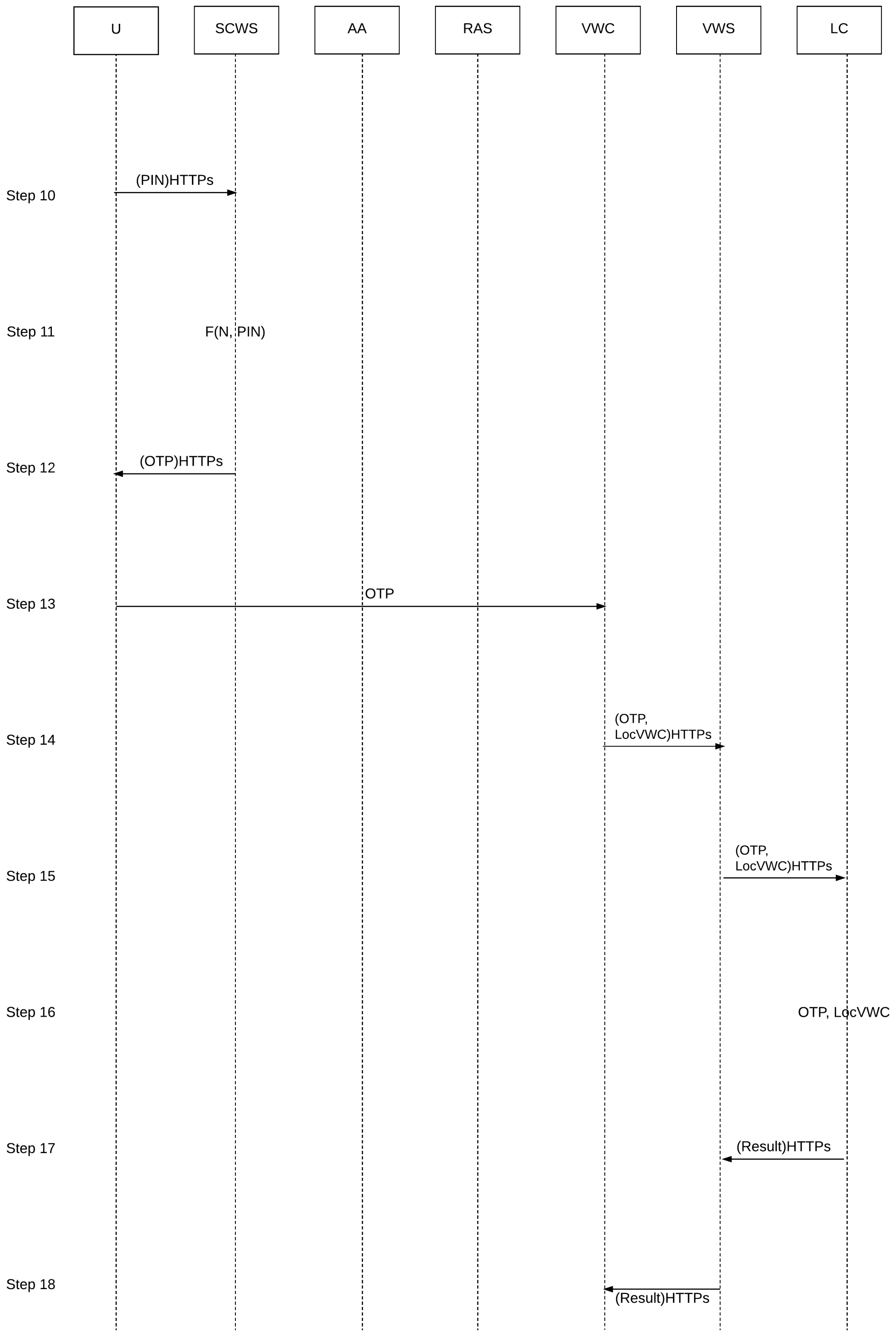


Figure 6.5: Virtual World Protocol Flow (Steps 10-18)

Malware on the phone: If phone malware steals the nonce from the SIM, it will still need the PIN number to generate the full OTP. If the malware is able to retrieve the

OTP after this is created on the phone and/or is able to detect the phone's location, the attacker has to pass the OTP and the location information to a VWC before the legitimate user does so, which may represent a timing challenge. Malware on the phone alone, does not have the necessary potential to become a threat, since it also needs a PC client under the attacker's control to succeed with the attack.

Security of the RAS: The RAS is defined as a trusted entity in the assumptions, and therefore it is hard to attack. The RAS could be considered a single point of failure, but as it is owned/operated by an MNO or a trusted third party, it should have restricted and controlled physical access so any unauthorised usage will be difficult. In this proposal, it is used as a mere conduit between the VW server and the SCWS, and does not store any secrets. The channel between the RAS and the SCWS is protected by HTTPS, and therefore has typical secure channel properties.

The OTP and the use of the PIN: The idea behind recreating the OTP is to mitigate the risk of mobile trojans stealing the PIN (as is the case with malware that is able to steal SMS messages that are sent as part of the authentication process as a second factor). The OTP will also not be created unless the user provides the PIN. So even if the malware steals N (which in any case is transfered over HTTPS, so there is another level of protection), it will not be able to recreate the OTP, unless it knows the PIN. The PIN is also useful in case the phone is stolen. In that instance the new holder of the phone will not be able to recreate OTPs unless she knows the PIN of the original user. Other OTP generators (such as Google Authenticator) do not offer such functionality (once someone gets hold of the phone there is nothing to stop them accessing and use the OTP application) or even if they offer (as is the case with MobilePASS+ from Gemalto), they do not run inside the tamper resistant environment of the SIM, thus they are vulnerable to attacks from malicious applications running on the phone.

6.4.2 Virtual World Back End Server

There is a generic attack which this protocol does not seek to address. If VW servers are compromised, this protocol will not protect the user or the VW developers. Malicious entities can get full control over the back end of the VW, and a user/avatar can then be impersonated. If the login server is compromised, the entire procedure fails and the malicious party can produce OTPs at will.

6.4.3 VW Client and Client PC

VWC and PC Security: The VW client is the easiest target to attack. In current VW login procedures, if a malicious user steals a user ID/password and downloads the appropriate VWC, a genuine user can be impersonated. However, in this proposed system, stealing the user ID is not enough to complete the login, as the user's SCWS/-phone is also required. The advantage of this proposal compared to current practices is that if a static password is stolen, it can be used until the genuine user changes it, whereas with OTPs, an attacker gains access to one session only. Within this context, a reasonable compromise between security and usability is achieved. A highly motivated and knowledgeable attacker could still target a particular user/avatar, but the proposal sets more challenges for them to overcome.

Physical compromise of the VWC: If there is a physical attack on the VWC equipment i.e. the computer is stolen, the malicious user will not be able to login into the VW system as the real authentication secret is sent to the phone. If the phone is stolen in addition to the computer then the attacker would also have to input the PIN on the phone to retrieve the OTP information from the SCWS, or physically attack the SCWS/SIM card.

Malware on the client PC: If there is malware on the client computer e.g. a keylogger which records the OTP as it is input by the user [98], it will not gain any advantage, as the short validity period of the OTP means that authentication will probably complete before the malicious entity can mount an attack. If the malware is able to perform a DoS attack against the VWC, so that the user will not be able to submit the OTP, it still needs to spoof the IP address of the user. A certificate is also used to bind the VW client to the VW server, so malware would also have to steal or replicate this certificate and the corresponding private key for a successful attack. Although these attacks are possible, using the SCWS in conjunction with the PC will increase the complexity faced by malicious entities and may discourage less-motivated/unskilled attackers.

Mutual authentication: In order to provide mutual authentication it is assumed that the VW client is given a VW certificate during registration. We acknowledge that issuing a certificate to all these VW clients may be a cumbersome task. Authenticating a client using a certificate is generally considered a best practice. Our protocol can function without the client certificate, however there will be one less layer of protection. We also acknowledge that even with the certificate in place, a malicious entity can possibly reverse engineer an application and identify the location of the certificates and/or the accompanying private keys or even develop malicious programs that could attack the application and steal these credentials. However, due to the fact that there exist no application that is impenetrable, we anticipate that the extra layers of security can provide a certain level of protection to the user.

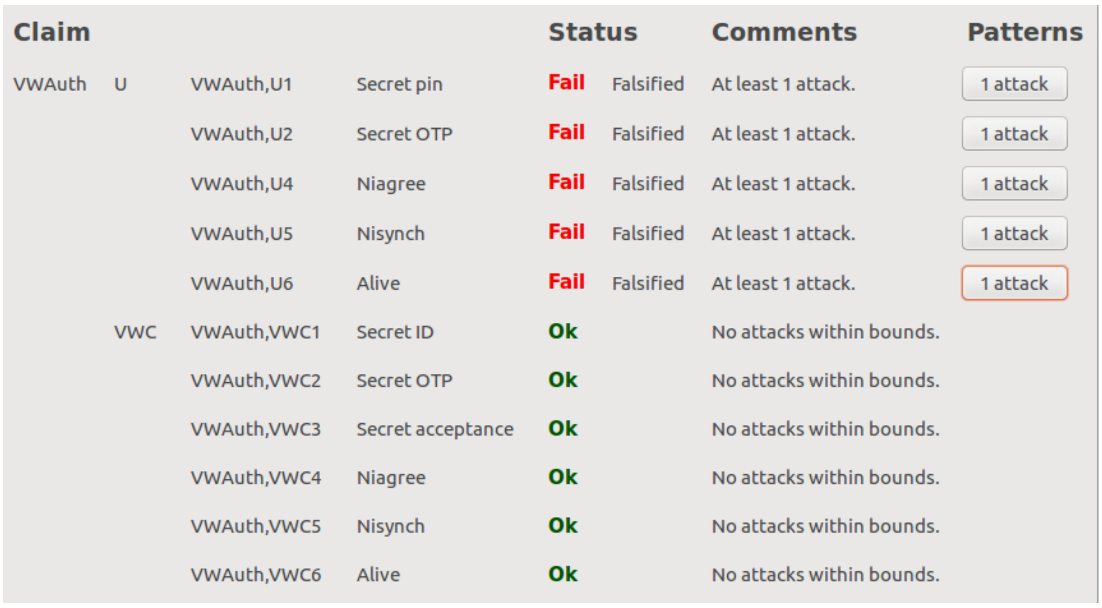
Data in transit: Data in transit is protected by HTTPS at all stages: from phone browser to SCWS, RAS to SCWS and between the VWC and VWS. HTTPS is the de facto protocol used whenever security is needed on the Web. There are recent attacks reported against HTTPS [117], but HTTPS provides reasonable protection against eavesdropping and man-in-the-middle attacks.

Replay attacks: Replay attacks do not work as a particular OTP is used only once and is valid for a short time period.

Geolocation: IP spoofing [135] can also compromise the login process. IP geolocation can be spoofed by a motivated attacker [136], but including it in the protocol adds an additional obstacle for an attacker. Even though IP spoofing is not extremely difficult, the attacker also needs to know the location of the user/phone at the exact time of the attack and simultaneously prevent the user from submitting the OTP through the genuine client. Geolocation checking is aimed against distant attackers, because if an attack is mounted within the genuine user's region it can be easily bypassed. Depending also on the risk appetite of the VW owners, the location of the phone may be used as an indicator to proceed with extra levels of protection. The VWS can store all locations that the user authenticated from and use it to identify whether to fall back to extra levels of protection, e.g. if a user never logged in from a specific location then the user is asked for the OTP whereas if this is a known location, then the user is not asked to provide the OTP, thus steps 9-16 can be omitted. However in this thesis we do not consider this option although such functionality may exist.

6.5 Formal Analysis of the Protocol

This section describes the formal/verification analysis of the protocol presented in this chapter. The basic entities are U (User), VWC (Virtual World Client), VWS (Virtual World Server), LS (Login Server), RAS (Remote Admin Server) and the SCWS. Next to them are the claims for each of these entities and possible attacks are defined in the last two columns. Figures 6.6, 6.7 and 6.8 present the analysis of the protocol by Scyther. The relevant code can be found in Appendix C. Scyther identified two attacks, as shown in Figure 6.9.
The first attack assumes that there is a malicious redirection to a fake SCWS. While this is not impossible, i.e. the user being tricked to submit the PIN to a phishing website appearing to be the one served by the SCWS, we anticipate that the attack will not offer any advantages to the attacker, since the PIN alone cannot create the OTP that will allow the attacker to login as the user. The attacker will need the nonce too, which is delivered to the SCWS over a secure protocol (HTTPS). So, unless the attacker is able to break the HTTPS connection, then the PIN alone cannot be used to successfully login.



Claim				Status		Comments	Patterns
VWAuth	U	VWAuth,U1	Secret pin	Fail	Falsified	At least 1 attack.	1 attack
		VWAuth,U2	Secret OTP	Fail	Falsified	At least 1 attack.	1 attack
		VWAuth,U4	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		VWAuth,U5	Nisynch	Fail	Falsified	At least 1 attack.	1 attack
		VWAuth,U6	Alive	Fail	Falsified	At least 1 attack.	1 attack
	VWC	VWAuth,VWC1	Secret ID	Ok		No attacks within bounds.	
		VWAuth,VWC2	Secret OTP	Ok		No attacks within bounds.	
		VWAuth,VWC3	Secret acceptance	Ok		No attacks within bounds.	
		VWAuth,VWC4	Niagree	Ok		No attacks within bounds.	
		VWAuth,VWC5	Nisynch	Ok		No attacks within bounds.	
		VWAuth,VWC6	Alive	Ok		No attacks within bounds.	

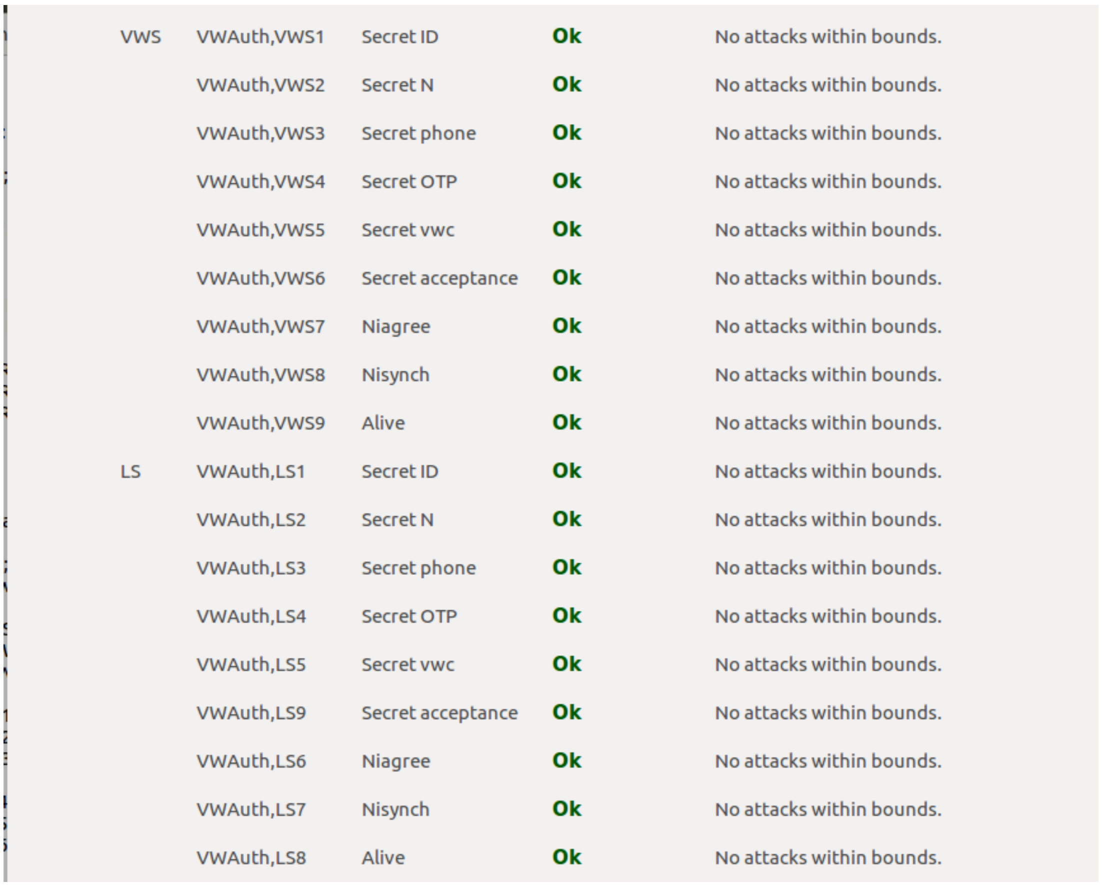
Figure 6.6: Virtual World Protocol Formal Analysis (1/3)

The second attack is just a false positive. The tool assumed that "Secret" is an entity that the user is sending the PIN and the OTP to, whereas the intention of using "Secret" in the code was to establish both the PIN and the OTP as two secret values known to the user.

6.6 Conclusion of the Chapter

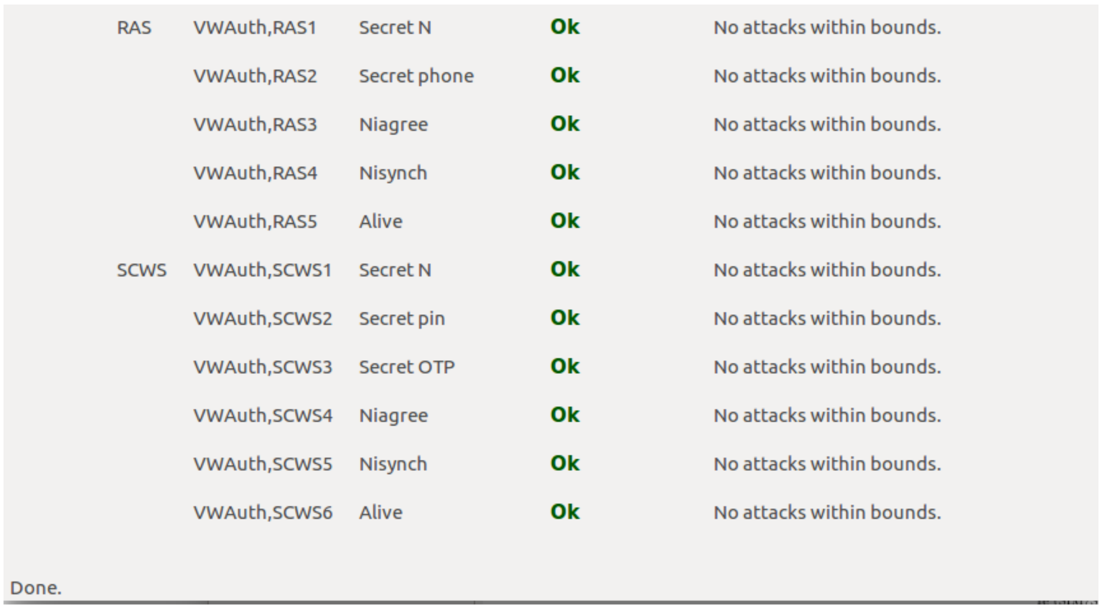
This chapter depicted a way of using the SCWS during the authentication process on the Web and more specifically during a login to a VW. An OTP creation inside the SIM card along with geolocation services, provide a certain level of assurance for a safe login process.

In the next chapter, the idea of using the SCWS to secure the Web will be expanded to the so-called Web of Things (WoT). The WoT is a subcategory of the Internet of Things (IoT) that aims at using web technologies within the realm of the IoT, mainly helping defeat interoperability issues. However, security is still being left aside, in favor of dealing with these interoperability issues first. Unfortunately, this is creating a risky position and this thesis aims to help establish an improved position in terms of security and more specifically tries to defeat authentication, to the WoT, issues.



VWS	VWAuth,VWS1	Secret ID	Ok	No attacks within bounds.
	VWAuth,VWS2	Secret N	Ok	No attacks within bounds.
	VWAuth,VWS3	Secret phone	Ok	No attacks within bounds.
	VWAuth,VWS4	Secret OTP	Ok	No attacks within bounds.
	VWAuth,VWS5	Secret vwc	Ok	No attacks within bounds.
	VWAuth,VWS6	Secret acceptance	Ok	No attacks within bounds.
	VWAuth,VWS7	Niagree	Ok	No attacks within bounds.
	VWAuth,VWS8	Nisynch	Ok	No attacks within bounds.
	VWAuth,VWS9	Alive	Ok	No attacks within bounds.
LS	VWAuth,LS1	Secret ID	Ok	No attacks within bounds.
	VWAuth,LS2	Secret N	Ok	No attacks within bounds.
	VWAuth,LS3	Secret phone	Ok	No attacks within bounds.
	VWAuth,LS4	Secret OTP	Ok	No attacks within bounds.
	VWAuth,LS5	Secret vwc	Ok	No attacks within bounds.
	VWAuth,LS9	Secret acceptance	Ok	No attacks within bounds.
	VWAuth,LS6	Niagree	Ok	No attacks within bounds.
	VWAuth,LS7	Nisynch	Ok	No attacks within bounds.
	VWAuth,LS8	Alive	Ok	No attacks within bounds.

Figure 6.7: Virtual World Protocol Formal Analysis (2/3)



RAS	VWAuth,RAS1	Secret N	Ok	No attacks within bounds.
	VWAuth,RAS2	Secret phone	Ok	No attacks within bounds.
	VWAuth,RAS3	Niagree	Ok	No attacks within bounds.
	VWAuth,RAS4	Nisynch	Ok	No attacks within bounds.
	VWAuth,RAS5	Alive	Ok	No attacks within bounds.
SCWS	VWAuth,SCWS1	Secret N	Ok	No attacks within bounds.
	VWAuth,SCWS2	Secret pin	Ok	No attacks within bounds.
	VWAuth,SCWS3	Secret OTP	Ok	No attacks within bounds.
	VWAuth,SCWS4	Niagree	Ok	No attacks within bounds.
	VWAuth,SCWS5	Nisynch	Ok	No attacks within bounds.
	VWAuth,SCWS6	Alive	Ok	No attacks within bounds.

Done.

Figure 6.8: Virtual World Protocol Formal Analysis (3/3)

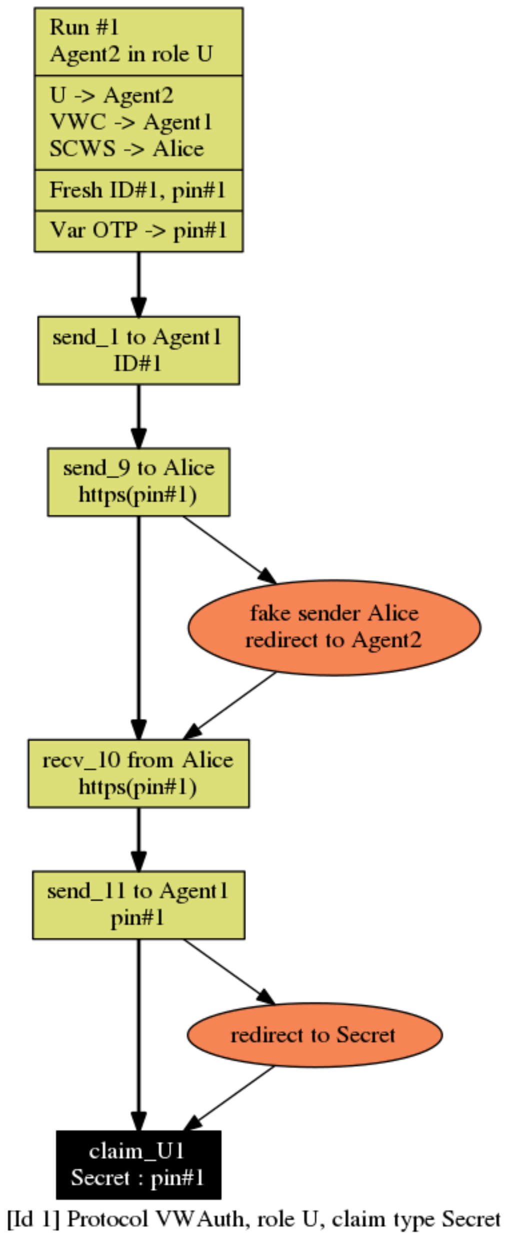


Figure 6.9: Virtual World Protocol Attack

Chapter 7

A Smart Card Web Server in the Web of Things

Contents

This chapter is based on the paper *A Smart Card Web Server in the Web of Things*, SAI Intelligent Systems Conference 2016 September 21-22, 2016, London, UK.

In the previous chapter we presented a way to allow the authentication of users in a virtual world environment through the use of a phone enabled with a SCWS. Up until now, the SCWS was used with the assumption that it is hosted on a phone (or additionally with a conventional credit/debit card as we have seen in Chapter 5). This chapter will extend its use to allow the SCWS to function in the wider Internet of Things and more specifically a subset of the Internet of Things, the Web of Things (WoT). This will be accomplished through the use of the so-called 'Security Modules', i.e. smart card chips that host a SCWS. These modules will be deployed throughout a location and will provide content to authorised entities in a secure and standardised way. Access to the modules will be given to these entities, after they have authenticated using a single sign-on solution that is also based on the use of a SCWS.

7.1 Introduction

The World Wide Web (WWW) is an almost ubiquitous technology. Since its introduction, it has changed the way we communicate, shop or even socialize and is now considered a vital part of everyday life. In recent years there has also been work towards technology that aims to connect everything in a communications network, the so-called 'Internet of Things (IoT)'. The IoT promises integration of all things in one large network, whether they be smart phones, computers, sensors, cars, meters, household appliances or tagged objects. The goal is not just communication, but widespread interaction to provide new and/or improved services.

Unfortunately, interoperability problems are currently inhibiting progress towards the IoT vision, resulting in isolated islands of connectivity using proprietary protocols. One approach to this problem is to create a 'Web of Things (WoT)', in which standardised web protocols are used to achieve the widespread connectivity required by IoT. Core to the WoT approach is the fact that TCP/IP stacks can be accommodated in devices with relatively poor computing power and memory. Tiny web servers inside WoT devices can be accessed by standard browsers and associated protocols.

However, the WoT approach comes with security concerns. The emphasis to date has been primarily on functionality, with security as an add-on. Using web protocols gives assurance about protocol design, but does not address attacks on implementation security, which is extremely important with so many accessible deployed nodes.

This chapter proposes a way to deal with such problems, by leveraging standardisation from mobile communications in the form of the SCWS that takes advantage of the tamper resistant (attack resistant) web server capability of a SIM card. For WoT we propose a strongly tamper resistant SCWS security module based on a Java Card v3.0 Connected Edition chip, supporting a TCIP/IP protocol stack. Finally, the chapter also proposes a way for local WoT single sign-on using a smartphone with a SIM card hosting a standardised SCWS.

The rest of the chapter is structured as follows: Section 7.2 provides background to relevant work in this area, Section 7.3 gives information on an alternative solution on WoT authentication and compares it against the proposal in this chapter, while Section 7.4 introduces our proposal. Example use cases and assumptions are described in Section 7.5, Section 7.6 outlines the main characteristics of the proposed architecture, Section 7.7 provides calculations for the execution times of the protocol, while Section 7.8 considers how it would be used for local WoT single sign-on. Section 7.9 gives a preliminary security analysis of the proposal and also discusses its advantages and disadvantages. Section 7.10 concludes the chapter.

7.2 Related Work

There are various definitions for the Internet of Things and perhaps the first time that a definition was given in public was during a presentation at Procter & Gamble [137]. However, if we interpret IoT using our own words, we could say that the IoT is the idea of everyday objects (fridges, ovens, cars, etc.) having network connectivity so that they can communicate with each other. These objects may have a range of functional abilities and be 'aware' of their own capabilities and in some cases of their environment, and can act individually (e.g. fridge) or as a group to provide a useful service (e.g. sensors in a house). The objects can range from everyday items with tags and/or sensors to major devices such as cars or houses. Recent surveys estimate the number of these objects to reach many billions by 2020 [138].

The emergence of the IoT is motivating significant research. A useful survey describing the technologies behind the IoT, the main applications that can be derived from its use, as well as concerns and issues can be found in [139].

In more recent years the WoT, a subcategory of the IoT has also drawn the attention from the research community. In [140], [141] and [142] the authors describe how they can integrate devices into the Web using the Representational State Transfer (REST) architecture, giving an even more detailed presentation in [14]. Another presentation on WoT issues including technologies, security, privacy and implementation can be found in [143]. Implementation discussions can also be found in [144] where the authors describe a number of prototypes they created. In [145] the authors describe the use of a gateway in order to access sensor nodes that cannot use web protocols. In [146] the idea of using web technologies with home appliances for home automation is presented, while in [147] the researchers describe a framework for the WoT, illustrated by a prototype motion detector system.

The above solutions define the idea around the WoT, however none of them addresses the issue of authentication in the WoT realm and how to enable secure communications between entities participating in the WoT. With the solution presented in this chapter we aim to propose an answer to this problem through a single sign-on (SSO) solution and the use of established protocols like TLS. We acknowledge that TLS is not suitable for sensors and miniature tags, but the devices that we are considering in this chapter have significant power to support protocols that require bigger amounts of power and our aim is to research the feasibility of using such protocols in the realm of the WoT.

For this proposal we will use the SCWS to serve content to clients running on a reader that interacts with the SIM card (usually the client is the browser on the mobile phone). For this chapter we are extending the idea of the SCWS by integrating it with objects that participate in the WoT - an SCWS in this case is called a 'security module'.

7.3 Web of Things Security

A community of developers that are behind the idea of the Web of Things, created a GitHub project to assist collaboration. It can be found in [148], but unfortunately it does not seem very active. The GitHub project contains a part about the security of the WoT and the administrators of the project.

It is interesting to read that they too recognise the necessity for security in the WoT, but acknowledge that most of the things lack such security.

The authors of the web page describe a number of possible attacks against the things, such as capture attacks where the attacker aims at compromising a thing, in order to steal information, disrupt it or manipulate it, creating safety risks where the crooks aim at exploiting a thing in order to threaten the physical security of a heart pump or a camera and even be used to attack other systems, by facilitating a DDoS attack.

7.3.1 Authentication

Authentication in the WoT, is happening through the use of Public Key Infrastructure (PKI). All parties are identified by their public key, humans and things alike.

To manage this functionality, there exists a security agent software module. This module is either part of the thing (if it is powerful enough) or resides on an IoT gateway whenever the things have very low resources and cannot implement the necessary TCP/IP stack and support public key cryptography operations. The agent is responsible for authentication, authorisation and integrity. The agent creates a private/public key pair for each of the things and all participating entities publish their public keys to the network, so that they can be searchable by other entities. The system is also enforcing access control lists to identify the level of access on each of the things.

The authors then describe the algorithms that can be used and propose the use of

Eliptic Curve Cryptography (ECC), since it is more suitable for these devices, after a comparison against an RSA implementation. Finally, they provide a UML diagram that describes the whole architecture.

The use of private/public key is important for our solution as well, so at this level the proposal in this thesis and the proposal of the authors is similar. However, our solution relies on the already proven TLS protocol, whilst the other solution is unclear on this.

Another important aspect is that they do not define, where will the private keys of the users going to be stored. Every system that relies in a public/private key needs a secure hosting environment for the private keys of all entities. It is not clear in that proposal, where will all these keys going to be stored, and although an assumption is that some of the things may allow for a secure hosting environment, they do not define something similar for humans.

Our solution uses a security module for all entities. The security module can be attached on the electronic assembly, be a smart card or simply reside inside the (U)SIM card in a user's phone. The outcome is that all authentication credentials (public/private keys and PINs/passwords) are securely stored.

7.4 Proposal Overview

Our proposal is built around the use of strongly tamper resistant security modules embedded in all communicating objects that collectively provide the security foundation of the entire WoT. Fundamental to this is the ability for entities to securely authenticate each other prior to transactions. A security evaluated smart card chip supporting Java Card v3.0 Connected Edition functionality is used as our security module. We chose this chip, as it can be integrated in an environment where secure access and storage of information is needed, but at the same time is easily accessible to external entities (readers, phones) without the need for proprietary protocols. Communication may take place over TCP/IP and, so standard HTTP clients may be used, notably web browsers. Additionally, Java is one of the most popular programming languages and the required development would be very similar to application development on conventional Java smart cards.

To briefly recap we are proposing an attack resistant chip, hosting an SCWS that can serve (via the WoT context), trusted web content via standardised web protocols and is relatively simple to program and manage. For the rest of the chapter, these tamper resistant SCWS chips will be called security modules. For the communication channel we chose to examine Bluetooth Low Energy and Wi-Fi Direct, although other candidates included Near Field Communication (NFC) and cellular. Bluetooth LE is the main candidate as its requirements in power are minimal compared to other technologies.

We have to note that our security modules are in general much more powerful than traditional things. So, the overall estimates and assumptions of this chapter will not apply to an environment where predominantly lightweight sensors/tags are used. In our use case the smart card chips can be attached to an electronic assembly and then be integrated to items of higher value, thus allowing for better performance, safer storage and execution times and easier access from readers. This electronic assembly with the security module can then be attached to items of relatively high value, e.g. a painting or a large box full of items of smaller value, thus making our solution not suitable for use with tags or minimal sensors.

7.5 Example Use Cases

An example use case could be a product logistics scenario and in particular the interaction of an employee's smartphone with the security modules within a warehouse. The employee can authenticate to his phone's SIM card (this SCWS will be called 'local server' for the rest of the text) and then be able to extend this authentication to other devices with SCWS capability (offered by the security module) in order to access data and functionality spread throughout the local area. We have to note though that the use of a smartphone is not mandatory and any other reader can be used instead - the only requirement is that the reader should be able to communicate over TCP/IP and offers a Graphical User Interface (GUI) (like a web browser).

Another use case could be a logistics warehouse, containing a number of crates, where each of the crates has attached one of these security modules. The modules host all necessary information about the contents of the crates and this information is accessible to only legitimate employees and without the need for Internet (or other) connection.

A third use case can be when these modules are attached to items of high value, so for example if an item needs to be transferred from one location to the other, the details of the recipient (address, phone, etc.) can be accessible to the transport employee in a secure way, without any further equipment and/or infrastructure needed (like Internet connection).

7.5.1 Assumptions

For our proposal to work properly we have to consider the following assumptions:

1. The SIM cards cannot be directly controlled by any other entity besides the MNO. Thus we assume that the MNO is trusted and that it allows access to the SIM card (for servlets that will be used by the SCWS to run inside the local server).

2. We do not consider managerial issues in this chapter, but rather we anticipate that there is a secure procedure for the development, deployment and revocation of certificates and key pairs on the various phones.

3. As mentioned above, we assume that the security modules are installed on items that have a significant value, e.g. a fridge, a painting, etc. or any other expensive item and they are not used with tags or other minimal sensors.

4. We assume that the security module receives accurate external time from an external timer found on the electronic assembly.

7.6 Architecture/Protocol

Our tamper resistant SCWS chips may be deployed to provide authenticated access to sensitive information about the products found inside a warehouse. We rely on the attack resistance of these security modules to provide assurance that the chip's stored data, access control and functionality will remain as intended. Furthermore any additional application dependent functionality/data hosted on the security module is expected to be simple enough to be rigorously security evaluated e.g. against common criteria. Since the security module is going to be serving sensitive information only to trusted parties, there must be a way of authenticating entities requesting to access this information - the solution should be able to fulfill the requirements described below.

1. The local server should authenticate the user[1].

2. The security module should authenticate the local server and vice versa.

A high level diagram of our proposal can be seen in Figure 7.1.

In principle a user could be authenticated directly to all security modules via a unique password or a PIN that is stored in each of these modules, although this would rapidly become impractical as the number of modules and users grow. Our proposal assumes that each user has a single PIN/password. This PIN/password allows the user to authenticate to the SIM card local server and the authentication result is later used for authenticating with security modules located around the area. A security module will accept the local authentication result, since it trusts the tamper resistant local server that authenticated the user in the first place. Procedures would be used to ensure that

[1]For a more complicated solution there may be mutual authentication between the local server and the user's browser through the exchange of certificates. We will not consider this option for this chapter.

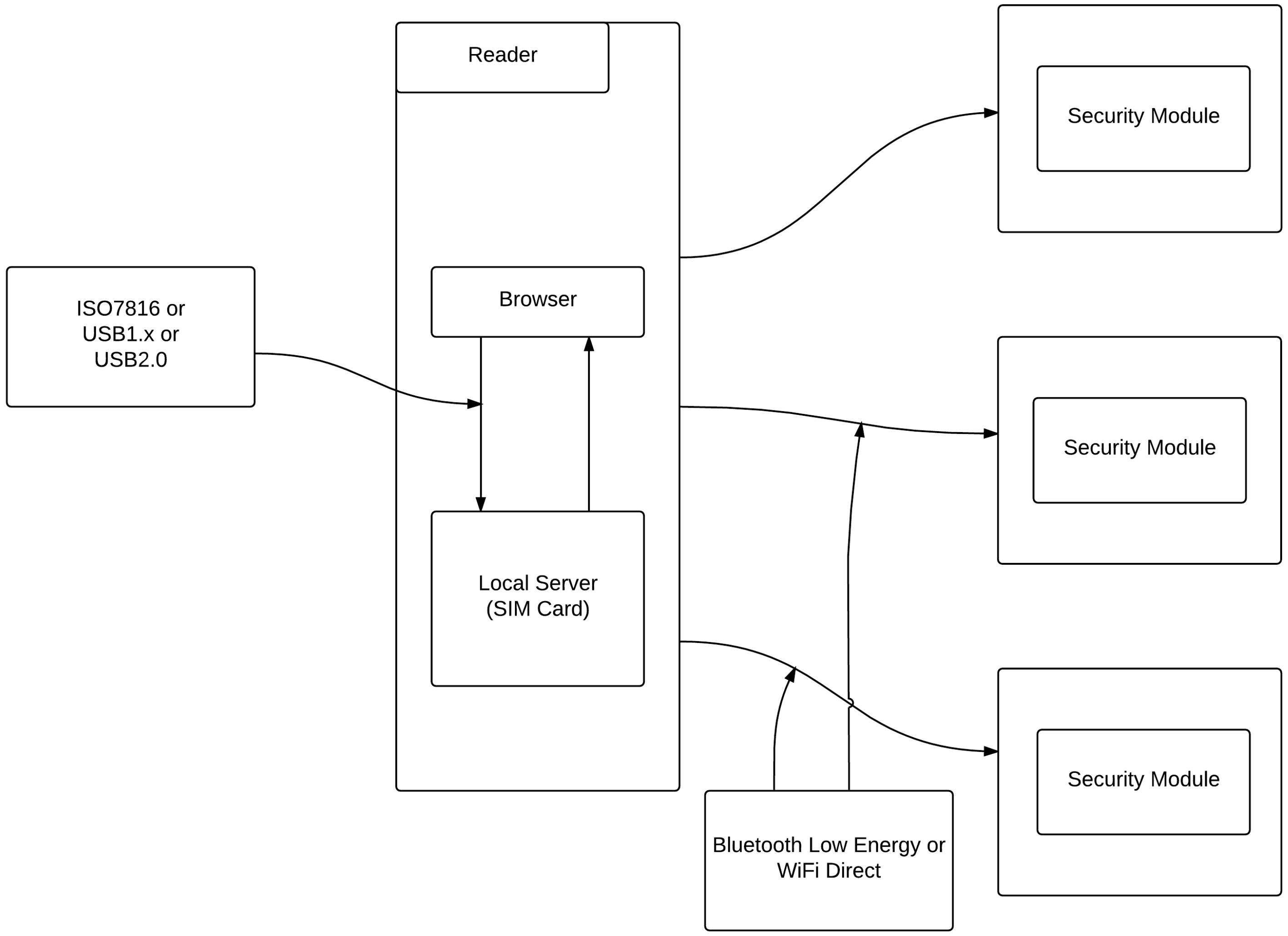


Figure 7.1: High Level Diagram

the user PIN/password was chosen, stored and updated in accordance with security best practice. The local server will create an authentication token that will pass over to the security module for the latter to provide access for a specific duration of time. The authentication token will have the following attributes:

- An attribute specifying the level of access granted by the authentication token (represented as L in Figures 7.2 and 7.3), e.g. administrative or standard access.
- An expiration date/time, represented as T.
- A value provided that will identify a unique SIM card, so that the authentication token will not provide access to other SIM cards. This value can be the Integrated Circuit Card Identifier (ICCID) that uniquely identifies a SIM card, represented as I.

Thus, an authentication token could be represented as the concatenation of the above attributes $(L||T||I)$.
The above can be seen in Figures 7.2 and 7.3.
We can summarise the authentication process in the following steps (Figure 7.2 represents steps 1 to 6, while Figure 7.3 represents steps 7 to 12):

1. The user starts the web client (browser) on the reader and starts negotiating a TLS connection to the local server (this is the first TLS handshake) - for TLS see [149].
2. The local server prompts the user for her password (or PIN). This is the 'PasswordRequest' HTML page.
3. The user provides the password/PIN to the local server.
4. The local server calculates the hash of the user's password and compares it to a pre-stored value.
5. The user is notified about the result. This is the 'PasswordResult' HTML page.
6. If the password/PIN is correct, the local server creates an authentication token, signs it and stores both inside the SIM card. The signature is created by the private key that is part of the key pair already present in the SIM card where the local server is host and can be later used to verify the integrity and authenticity of the token.
7. The user initiates a wireless connection with the security module using her reader. The two entities (local server and security module) negotiate a TLS connection.
8. After the second handshake is finished, the security module asks for the authentication token, from the local server. This is the 'AuthTokenRequest'.

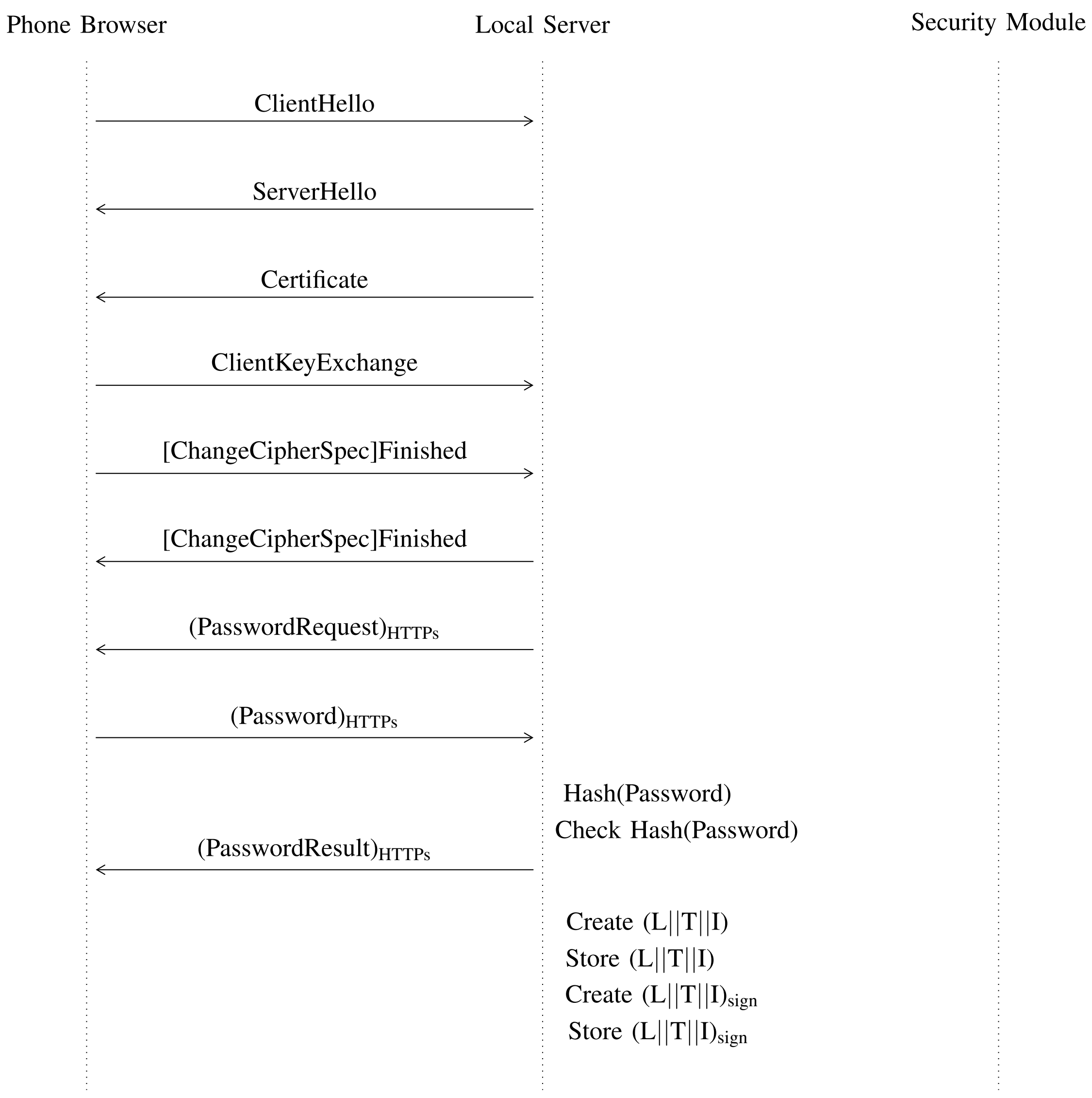


Figure 7.2: User Authentication and Token Generation

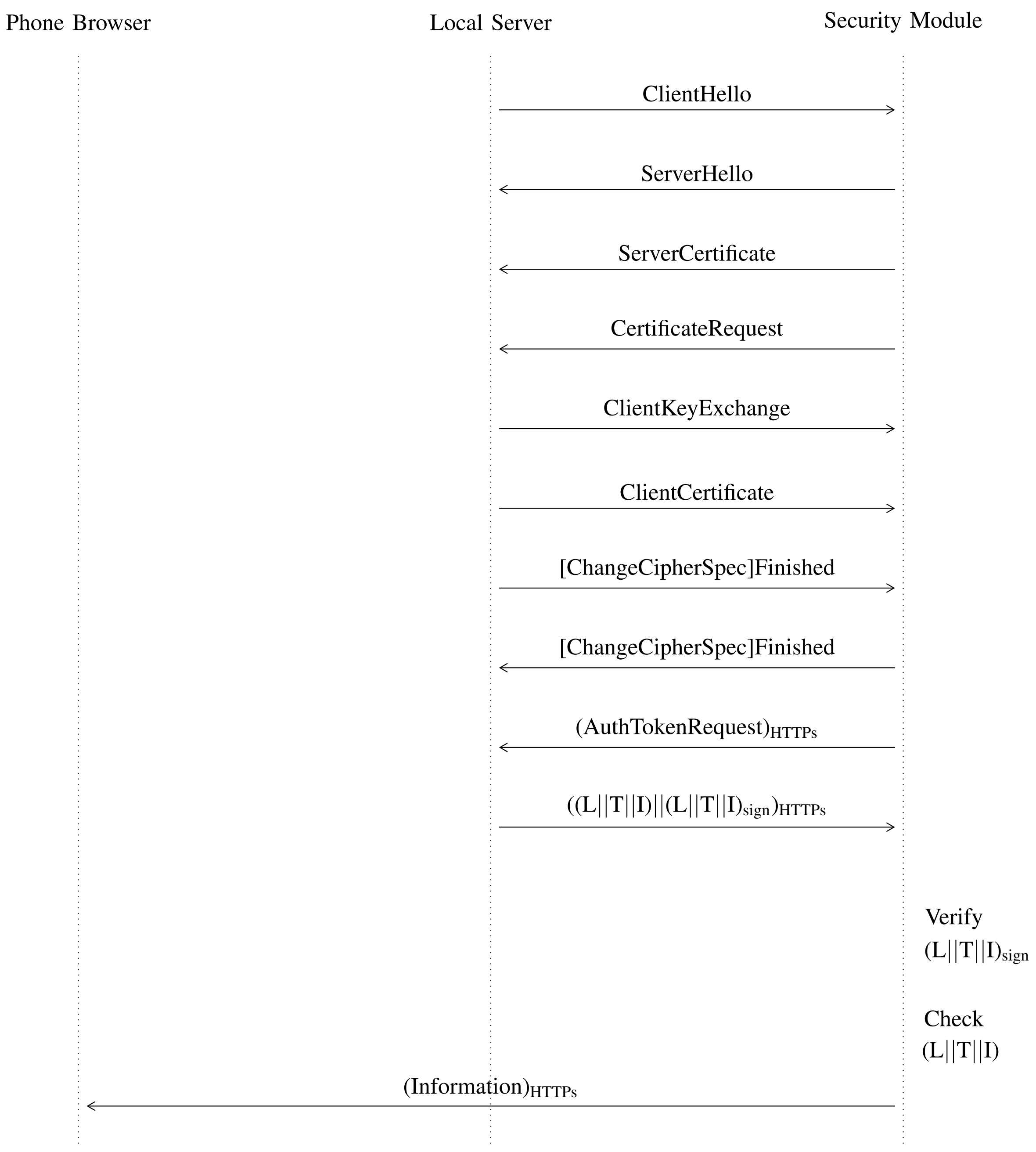

Phone Browser
Local Server
Security Module
ClientHello
ServerHello
ServerCertificate
CertificateRequest
ClientKeyExchange
ClientCertificate
[ChangeCipherSpec]Finished
[ChangeCipherSpec]Finished
$(\text{AuthTokenRequest})_{\text{HTTPs}}$
$((L||T||I)||(L||T||I)_{\text{sign}})_{\text{HTTPs}}$
Verify $(L||T||I)_{\text{sign}}$
Check $(L||T||I)$
$(\text{Information})_{\text{HTTPs}}$

Figure 7.3: Object Authentication and Access

9. The authentication token and the accompanying signature are transferred to the security module over the established wireless connection.
10. The security module checks the validity of the new authentication token, i.e. makes sure that the token has not expired. It also verifies the signature of the token using the public key of the local server that it received during the TLS handshake. Note: The certificate may already exist on the security module, so there is no need to be transfered during the handshake. However this is a case that will not be considered in this thesis.
11. If both are legitimate, then the security module grants access to the user to the information stored on it. If not, then the token and the signature are discarded and the user is notified.
12. The requested information is sent to the browser over the established connection.

We have to note, that the first part of the protocol can be avoided in case the token is still valid i.e. the user does not need to provide her PIN/password every time, but rather the process can proceed to the second part directly. This check is taking place inside the SIM card/local server - so the user needs to resubmit the PIN/password only when the token stored inside the SIM card has expired. The trust relationship between the local server and the security module means that the latter will accept the token's validity.

By always submitting the authentication token during the second part of the protocol, we ensure that the security module always gets and checks the freshest available token. Between a successful authentication attempt and a subsequent one, the user may have re-authenticated to the local server (thus expanding the validity of the authentication token) or her authentication level may have changed, so that now she may access less or more information. Furthermore in the case of a lost or stolen reader, the authentication token could be erased from the SIM card/local server, so that the non-authorized entity that may try using it to access a security module, will have to re-authenticate (in order for a new token to be created). However, the malicious entity will not know the PIN/password, thus adding another layer of defense for our solution. If the token had to be stored in each of the security modules that the user requested access to, then the tokens would have to be erased from all these modules to block malicious users.

7.7 Process Duration Time Estimation

For the purpose of calculating the overall time needed for the whole process to finish and the user to start receiving information/data in her browser, we wished to consider a modern Java Card v3.0 Connected Edition. Unfortunately at the time of writing, these cards were not available for the study (thus limiting the practical part of our research) and for our estimated calculations we had to use figures based on Java Cards of the previous generation (v2.2). Our aforementioned calculations are based on results that we gathered during other experiments (for example time needed for an encryption of X bytes to occur), as well as on data that we received from a leading smart card chip manufacturer. This manufacturer provided real figures (for the most powerful of the smart cards that they produce) for a number of actions like RSA signing/verifying, SHA-256 hashing, etc. using a crypto co-processor (please see Table 7.1). For the communication between the browser and the local server, we assumed the use of three different standardised communication channels: conventional ISO 7816, USB1.x and USB2.0 [26]. For the (wireless) communication channel between the local server and the security module we used Bluetooth Low Energy and Wi-Fi Direct.

We separated our calculations into two phases: the first phase is between the browser and the local server (thus, the SCWS inside the SIM card), while the second phase is taking place between the reader/SIM card and the security module. As can be seen in Figures 7.2 and 7.3, the messages are presented with the handshake broken down to its basic parts. For each of these messages we calculated the time for the message to be transmitted and/or the time for the basic action to occur, such as an encryption of certain amount of data or the verification of a signature.

Table 7.1: Algorithm Speeds

Action	Algorithm	Timings
Symmetric Encryption/Decryption	AES-128	0.2ms/block
Asymmetric Signing	RSA-2048	150ms for 2048bits
Asymmetric Verification	RSA-2048	9ms for 2048bits
Hashing	SHA-256	0.5ms/block

The amount of data exchanged for the handshakes, the size of password, password hash and authentication token that were processed at each step can be found in Table 7.2 - please refer to Figures 7.2 and 7.3 for detailed information about the contents of each of the protocol steps. For messages exchanged over HTTPS we included a further overhead for the TLS headers. The page size represents the HTML page that was created by the local server and send to the phone's browser. The HTML page is represented as PasswordRequest which is a page created by the local server for the user to input her password and PasswordResult, where the user is informed about the password checking result.

Before running the calculations we anticipated that the bottleneck for the whole procedure might be the wireless interface between the reader and the security object and (if used) the conventional ISO 7816 interface between the phone's browser and the SIM card. Thus, we concentrated our attention to the comparison of the various available communication channels that we could use, i.e. ISO 7816, USB1x and USB2.0 for the communication between the SIM card and the browser ('wired' communication), and Bluetooth LE and Wi-Fi Direct for the 'wireless' interface.

Table 7.2: Amount of Data

Data	Amount (in bytes)
First TLS handshake	5149
Second TLS handshake	9399
Password	9
Password hash	32
Authentication token	36
HTML page (incl. TLS headers)	70

Times for data transit of all messages between the entities can be seen in Table 7.3 (calculations in milliseconds). This table does not include the processing time for actions on the data (can be found in Table 7.4), but the times depicted are the summary transit times for both parts of the protocol (e.g. transit over ISO 7816 for the first part of the protocol and transit over Bluetooth LE for the second part). Next to each of the options the reader can find the assumed communication speed for each. ISO 7816 has a broad range of communication speeds depending on the hardware and the vendor, so we selected an average figure. The same principle was applied to the rest. It must be noted that the theoretical speed for each of the interfaces is larger, but we chose to use a figure representing an average used/found in practice.

Table 7.3: Transit Times per Communication Channel in ms

Wired Protocol	Wireless Protocol	
	Bluetooth LE (0.1 Mbit/s)	Wi-Fi Direct (54 Mbit/s)
ISO 7816 (0.1 Mbit/s)	1967.04	1208.69
USB1.x (12 Mbit/s)	769.82	11.47
USB2.0 (80 Mbit/s)	761.27	2.92

Table 7.4 depicts the time needed for the most important calculations to finish. This processing includes the TLS handshake with the verification of certificate signatures, hashing, encryption/decryption of data to be transmitted over TLS and authentication token signature creation (we assume that both chips, in the SIM card and the security module, are of equivalent power).

Table 7.4: Calculations Times

Action	Size of data (in bytes)	Time (in ms)
Signature Verification	256	9.00
Data Hashing (per certificate)	1500	12
Auth.Token Signing	256	150
Data Encryption/Decryption (SIM card and Security Module)	52	0.8
Password Hashing (for 9 bytes password)	32	0.5

Table 7.5 depicts the overall time for each of the phases, i.e. it is the summary of all calculations to occur.

Table 7.5: Total Calculation Times per Phase

Phase	Time (in ms)
User Authentication and Token Generation	216.40
Object Authentication and Access	143.60

Table 7.6 gives the overall time needed for the whole process to conclude. Our estimated calculations show that the communication speed is dominating the duration of the process (compared to the processing), especially when the combination is for ISO 7816 along with Bluetooth LE (for the wireless interface between the security module and the reader).

We can see in Table 7.6 that the combination of Bluetooth LE with ISO 7816 is almost 7 times slower than the equivalent combination of Wi-Fi direct and USB2.0.

Table 7.6: Total Times in ms

Wired \ Wireless	Bluetooth LE	WiFi Direct
ISO 7816	2327.04	1568.69
USB1.x	1129.82	371.47
USB2.0	1121.27	362.92

7.8 Local Single Sign-On

At the beginning of the chapter we discussed about how our proposal provides a single sign-on (SSO) solution that will allow users to authenticate in a de-centralised fashion in their phones/readers and then be able to access information found on security modules. We now describe how this can be accomplished.

The idea of SSO is not new: in fact it is a well established technology and there are many, free and commercial, SSO product solutions. While products have different features, the basic idea behind SSO is that a user authenticates to just a single central server, which then handles the user's authentication to other participating servers. One of the most prominent of these solutions is OpenID [51] that is being used by a large number of websites and most recently, social media solutions like Facebook Connect allow users to authenticate anywhere using their social media account [150].

One of the most important protocols for SSO is the Kerberos protocol developed by MIT [151]. Kerberos is used with many solutions/platforms, the most prominent of which are the Windows OSes developed by Microsoft [152]. In a nutshell, Kerberos allows users and systems to obtain a 'ticket' from a Kerberos Key Distribution Center (KDC) and use this ticket as their credentials when required to access other entities within the network. These entities trust the Kerberos ticket credentials and so allow access from the users/systems that were previously authenticated by the KDC.

Our proposal uses ticket credentials, but is otherwise a different approach. The user is now the 'owner' of the authentication server, within the SCWS/local server of the SIM card. The credentials of the user are stored inside the SIM card (which is a tamper resistant device). The user authenticates to the local server, and the SCWS then creates a token that the user can use anytime to access a security module. An advantage of this approach is that there is no centralised equivalent to the KDC, so no single point of failure or target for Denial of Service or other hacking attacks. Of course, there should be a general trust relationship between all the entities used in the system and this can be achieved by a conventional X.509 certificate approach. Once the local server and the security module mutually authenticate, the security module retrieves the token stored in the SIM card and the user is authenticated if the token is valid and fresh.

7.9 Security Analysis

7.9.1 Advantages

We based our security module on a smart card chip because such chips can be strongly tamper/attack resistant. Modern high quality chips will have effective defenses against physical probing/modification, side-channel and fault attacks and can be independently evaluated against common criteria. In short, providing weak devices are avoided, we expect the module to protect the integrity and access of sensitive stored data and to function as per design/expectation, even when subjected to aggressive attack.

In any case no sensitive data is available without authentication and no sensitive transmissions are in clear. The same levels of protection are offered by the local server.

One of the biggest advantages in favor of our proposal is achieving interoperability, without compromising on the tamper resistant security of the solution. All involved entities communicate over TCP/IP channels, thus making them accessible from any platform that supports this protocol suite i.e. practically everything in the modern Internet. An attack resistant web server stored inside a security module (chip) implementing a TCP/IP stack means that is easily accessible using standardised protocols and can be incorporated within an WoT architecture that requires all the involved entities to be accessible over 'web' standards. The lack of need for special protocols and proprietary communication methods allows for easier integration in any environment, and application development is straightforward for anyone familiar with Java and/or smart card programming. The attack resistance of the security module and the equivalent attributes of the SIM card (where the initial authentication is taking place), protect the data and functional implementation (contained within the chip). Logical attacks (against the design) cannot be avoided, but the small attack surface of the web servers and the servlets involved, provide a certain assurance level that the programmers can develop robust and security verified code. Another important factor is that all communication is taking place over secure channels and more specifically using HTTPS. Attacks against HTTPS exist [153], however the vast majority of web servers throughout the Internet rely on HTTPS to secure communication to and from them. Until proven otherwise, HTTPS is still the most secure way to browse the Internet and as such should be acceptable for WoT.

Another potentially important factor (which we consider a strong privacy advantage of our proposal) is that the user's password/PIN stays within the local device and is checked by the local server (within the trusted SIM card). The PIN is exposed to the browser (when it is submitted by the user), however even if malware captures the password/PIN, a malicious user also has to be in possession of the exact SIM card in order for her to use it (because the password/PIN can be used with one SIM card only as it is stored in it and checked by it). So, even if an attacker steals the password/PIN, she will need not be able to use it effectively, unless she is in possession of the SIM card as well. Also, if the attacker steals the authentication token, she will not be able to use it, since the authentication token can be used by one SIM card only (identified by the ICCID). Furthermore, the attacker needs to be in possession of a X.509 certificate installed within the 'malicious' SIM card; this certificate must be signed by the CA authority that governs the entire system, which means that the attacker either is in possession of the private key that corresponds to an existing valid certificate or has her own certificate signed by the CA - the former is quite difficult to achieve and the latter should be forbidden by CA processes and controls.

The security modules contain all the necessary information about the product(s) that the user wants to access. Thus, a product can provide this information when tapped by an authentic user/phone, and without the need for on-line communication. This stand-alone ability of the chip is very important when an Internet connection is not present. Finally, a web enabled security module has enough processing power to run servlets and process data that it stores/receives. Other WoT entities like sensors possess limited (if any) processing power and in most cases they are just simple conduits pushing forward information to more capable entities.

In summary we can say that the main advantages of this proposal are:

1. Interoperability using standardised/proven open protocols.
2. Attack resistance using proven attack resistant devices.
3. Communication exclusively over secure channels via HTTPS.
4. Web protocols that allow the entities in this proposal to seamlessly form part of the WoT.
5. Stand-alone entities that may provide information without the need for a back end server/database nor a connection to the Internet.
6. De-centralised single sign-on, thus avoiding single points of failure.
7. PIN/passwords and authentication tokens are useless without the SIM storing them. Thus the attacker has to be in possession of the PIN/password or authentication token plus the SIM card in question, in order for her to get access to a security module.

7.9.2 Disadvantages

The importance of mutual authentication cannot be underestimated, however installing and maintaining a number of certificates with all the accompanying tasks (updating, revoking) could prove cumbersome. We anticipate that a process of updating the certificates will be down to policies mandated by the individual case/scenario with the attack resistance of the chips permitting a longer period of certificate validity, depending on the value of the information stored on the chips.

Another fact is the cost of the security modules. An assembly with an advanced smart card chip may be too expensive for use with ‘cheap’ tags/things. So, for our scenario we assumed the use of these objects with items of generally higher value. For example, an assembly with a security module can be attached on a painting, on a fridge or on a palette containing a large number of items. It can then store and protect information about these items (information for the painting or the contents of a box/palette) in a tamper resistant way protecting it from malicious entities.

7.10 Conclusion of the Chapter

This chapter presented the idea of extending the SCWS to the IoT/WoT area. This proposal uses the Smart Card Web Server inside a phone and in a smart card chip (security module) and allows for secure authentication of users to these security modules through the use of a single sign-on process. The SSO server is stored on the user’s phone (or any device that can support the existence of a smart card with an SCWS) and allows the user to authenticate to this server with a PIN/password and subsequently to authenticate to security modules that trust this SCWS.

The next chapter will conclude this thesis.

Chapter 8

Conclusion

The work that was presented in this thesis addressed the research question as declared in Chapter 1, i.e. the feasibility of an integration into the Web and the WoT of SCWS-enabled smart cards/SIM cards. This integration was demonstrated through four use cases that addressed four different security problems/concerns. This work depicted that this integration is feasible and that SCWS-enabled smart cards can help mitigate these concerns in an effective way. All four proposals included a well defined protocol that described the necessary entities that need to be involved in each one, the necessary steps that have to be taken and any security concerns that may arise.

The first solution as presented in Chapter 4 allows the distribution of the voting process in multiple phones. Each user will/can have a voting website hosted locally on her phone and she will be able to vote even without Internet connectivity. For an attacker to disrupt the voting process will need to attack all (or a large number) of these phones. Identifying which phone is going to be used for the voting procedure can be a difficult task on its own, which means the attacker needs to get hold of information about the participating phones and then to attack them. Even if the attacker manages to attack certain phones, not allowing the user to vote, there is always the alternative of the booth vote. DDoS attacks are thus becoming much more difficult to implement, since there is not a single point of failure and the voting process is practically distributed along a potentially large number of phones.

This thesis has also presented, in Chapter 5, a way for the improvement of the security of e-commerce transactions. Our proposal adds the Smart Card Web Server to the payment process, which can be used as a secure personal repository for cardholder data that is only revealed back to the issuer during payment transactions. The solution makes use of existing and proven protocols (HTTPS), access via standard browsers, single use virtual card numbers, and complemented by an application level security solution. Cardholder privacy should be improved and merchant requirements for PCI DSS should be avoided (or at least relaxed). As mentioned before, if a user used this solution with every Internet transaction, she would have never been required to remember or submit any sensitive cardholder data, thus enhancing the security of this information. Additionally, merchants may be required to comply with less PCI DSS requirements, thus making e-commerce processes easier and cheaper for them.

Chapter 6 presented a third proposal, i.e. a method for using the SCWS to enhance the login procedure of virtual worlds by using one-time passwords and location based checks. The use of standardised protocols (HTTPS, FAP) along with existing security hardware (SIM/SCWS) and secure communications, simplifies the design and operation of the proposal. The preliminary security analysis indicated that the proposal had promising capabilities to prevent unauthorised access to VWs. The advantage of our proposal compared to other authentication solutions employed by VWs is that it uses a second factor of authentication, along with a one-time password combined with geolocation capabilities. The OTP is recreated inside the SIM card, thus avoiding malware that may try to retrieve it and the use of the user's PIN that is used in the OTP calculation provides and extra layer of security, given the fact that the whole process runs inside the secure SIM environment. Even if the phone is stolen, OTPs cannot be created, unless the attacker is in possession of the PIN as well. Using this proposal mitigates the fears of phishing attacks and makes login process much more secure.

The final proposal was described in Chapter 7 and uses the Smart Card Web Server both in an attack resistant SIM card (for which it was initially standardised), but also in a similarly protected smart card chip (security module) embedded within an electronic assembly. The two entities communicate over a wireless communication channel mutually authenticating each other. This way the web server running inside the security module accepts the authentication results for the user that was previously authenticated from the SIM card local server. This SSO solution has the advantage of avoiding a single centralised SSO server as it distributes the authentication process over a number of trusted local servers found inside SIM cards. The entire solution works even when there is no network connectivity available, as everything is happening offline. The user still needs to remember a password/PIN, although capture of this credential is of little use without the corresponding SIM card.

8.1 Summary

As stated above, this thesis attempted to provide an answer to the question: how can we use a Java Card v3.0 Connected Edition with a Smart Card Web Server to enhance the security of web applications and to assist in a more secure authentication in the WoT? This research question was then ‘divided’ into four subquestions, with each subquestion dealing with a specific security threat affecting the Web or the WoT and demonstrated by equivalent, four use cases. Tables 8.1 and 8.2 summarise the findings of this thesis and whether our research objectives were met. As these tables depict, we believe that this thesis managed to successfully reply to the research question as was set in Chapter 1.

8.2 Future Work

The SCWS is a relatively new development in smart card technology, so not every phone will have the SCWS/SIM required for the solutions proposed in this thesis. In fact, at the time of writing a SCWS-enabled smart card was not easy to find and use. Unavoidably the thesis approaches the subject from a theoretical point of view. Future work can prove the feasibility of solutions presented in this thesis.

Regarding our first proposal, additional enhancements to the model could also be to allow one phone to process votes from multiple voters. Each voter would need personalised ballot forms on the SCWS, accessible to them only once they are suitably authenticated. Confirmation SMS messages from the voting authority could go to a nominated phone number, not necessarily back to the SCWS/SIM phone used to vote. Since the SCWS does not need to be online to process votes, the proposed SCWS voting method could potentially allow a mobile phone to be used as a portable voting booth in situations with no network connectivity or if certain phones cannot be used to vote because of an attack. Votes could be stored on the SCWS if there is sufficient storage capacity (or else encrypted on the phone), then uploaded to the e-voting system at a later stage, perhaps via a transfer point in a controlled voting kiosk area, or once connectivity is restored. This suggests that the proposed system could be used for voting in difficult, challenging and resource-constrained environments such as those existing in remote regions of the developing world. NFC phones and contactless smart card IDs could provide an alternative, easy to use method for authenticating voters, using the NFC phone as a smartcard reader. This requires an application on the untrusted mobile handset, but authentication credentials can be protected in transit through the phone by encryption. NFC allows the use of an identity token with a strong independent linkage to identity, such as a passport or government-issued identity card, e.g. as in [154]. It is intended that the proposed SCWS e-voting process will be implemented as a practical system on a range of phones and operating systems to enable testing of the robustness of the design, its usability, acceptability to voters, and its practicality in challenging infrastructure-poor situations.

The solution presented in Chapter 5, would be benefited by a practical implementation that will identify the speed of the whole process and the ease of use for the

Table 8.1: Summary of Objectives (Chapters 4 and 5)

General Objective	**Risk**	**Result of our Research**
Protection against (D)DoS attacks	(D)DoS attacks are becoming more powerful and at the same time more difficult and expensive to defend against. Meanwhile, the barrier of difficulty for launching such attacks is getting lower through the use of criminal services.	We believe that our objective was achieved. Our proposal demonstrated that by distributing a process (in this instance e-voting) to a number of ‘personal voting websites’, makes it much more difficult for malicious people to launch a (D)DoS attack, since they will need to target a large number of different websites, each one being hosted on a different phone. Identifying all these ‘personal voting websites’ is in itself a challenging task, let alone targetting all, or at least a large number of them, in order to affect the voting process.
Secure e-commerce transactions and easier PCI DSS compliance	The security of e-commerce transactions is threatened by malicious people that try to trick their victims into submitting their cardholder data to phishing websites or by attacking merchant shops and back end servers to steal cardholder information in large numbers. At the same time, PCI DSS compliance is a cumbersome and expensive task for all merchants.	We believe that our objective was achieved. Our proposal demonstrated that by using a smart card/SIM card that hosts a SCWS, the e/m-commerce experience can be enhanced for the end user, without having to sacrifice security. The user does not need to provide any cardholder data, thus minimising the risk of phishing attacks. Additionally, the use of single use credit card numbers, turns the cumbersome PCI DSS compliance into a much easier (and cheaper) task for merchants.

Table 8.2: Summary of Objectives (Chapters 6 and 7)

General Objective	**Risk**	**Result of our Research**
Safer authentication for web applications	End users are constantly falling victims to phishing attacks, while certain methods for second factors of authentication have their own weaknesses.	We believe that our objective was achieved. Our proposal demonstrated that by using a SIM card with a SCWS, a user can enjoy the security of a second factor of authentication offered directly by her SIM card, without having to install second factor authentication applications on the insecure part of the phone. At the same time, all secret values (PINs, nonces, OTPs) are transferred over secure channels, while the authentication process is further enhanced by the use of geolocation. The proposal used a virtual world as a use case, however it can be further expanded to more applications, especially for risk-based authentication.
Safer authentication in the WoT	Security in the WoT is not adequate and leaves the things insecure because of weak authentication practices.	We believe that we mainly achieved our objective. Our proposal allows for a much safer authentication in the WoT realm by using a single sign-on solution that is running on smart/SIM card that hosts a SCWS. Additionally, the security modules are able to participate in the authentication process and to offer web content to authorised entities only. Due to the fact that our research was limited to things with sufficient processing power, further research is needed to investigate the potential expansion of our proposal to less powerful things.

users. A very important piece of work will be to identify the feasibility of the integration of our solution with modern e-commerce and e-payment processes and websites. Every payment process should be straightforward for programmers to integrate with a website (possibly through the use of appropriate APIs and sample code), e.g. part of PayPal's success is due to the fact that integration of the PayPal button with any website is quite easy. It must be noted that our solution is not restricted to e-commerce transactions and future work may be applied to e-government and e-banking services.

A more detailed analysis will also be beneficial for the solution of Chapter 6 in order to fully understand its security properties and a practical implementation of the protocol would be useful to determine the speed of the various steps and how this would affect usability. Future work could also include an enhancement to the basic protocol to be used when a more advanced level of security is required (e.g. VW banking): this would involve a second OTP input to the SCWS, and sent to the VW via the RAS or to include a more risk-based approach, e.g. how will the protocol behave if the user is logging in from a location previously seen or if the location is new.

It will also be very interesting to see a practical implementation of the final proposal as presented in Chapter 7. The promising distributed SSO that was the core of the solution has to be examined from a user perspective, but also from a management point of view. The ease of managing a number of security modules in multiple scenarios and use cases will depict the workability of the solution. A very important fact will also be the calculation of duration of all phases of the solution to identify the speed of the protocol execution and especially the identification and confirmation of bottlenecks and of possible improvements considering more communication channels.

A more general ambition would be to influence the development of standards and specifications for the application of SCWS to IoT/WoT security, and thereby increase research and business interest in this promising technology, while also to extend our proposal to other Trusted Execution Environments (TEE) in the Internet of Things field.

Appendix A

Scyther Code (Chapter 4)

Scyther code for the security analysis of the protocol of Chapter 4.

```
usertype PublicKey;
usertype PrivateKey;
usertype Vote;
usertype String;
usertype Application;

secret https: Function;

protocol Voting(SIM, RAS) {

     role RAS {

        const PUBV: PublicKey;
        const PUBVA: PublicKey;
        const PKV: PrivateKey;
        const username: String;
        const password: String;
        const application: Application;

        var vote:Vote;

        send_1(RAS,SIM, https(PUBV, PUBVA, PKV));
        send_2(RAS,SIM,  https({username, password, application}PUBVA) );
        recv_3(SIM,RAS,  https({vote}PUBVA));

        claim_RAS1(RAS, Secret, PKV);
        claim_RAS2(RAS, Secret, username);
        claim_RAS3(RAS, Secret, password);
        claim_RAS4(RAS, Secret, application);
        claim_RAS5(RAS, Secret, application);

        claim_RAS6(RAS, Secret, vote);

        claim_RAS7(RAS, Niagree);
        claim_RAS8(RAS, Nisynch);
        claim_RAS9(RAS, Alive);
    }

     role SIM {

        const vote:Vote;

        var PUBV: PublicKey;
        var PUBVA: PublicKey;
        var PKV: PrivateKey;
        var username : String;
        var password: String;
        var application: Application;

        recv_1(RAS,SIM, https(PUBV, PUBVA, PKV));
        recv_2(RAS,SIM,  https({username, password, application}PUBVA) );
        send_3(SIM,RAS,  https({vote}PUBVA));

        claim_SIM1(SIM, Secret, PKV);
        claim_SIM2(SIM, Secret, username);
        claim_SIM3(SIM, Secret, password);
        claim_SIM4(SIM, Secret, application);
        claim_SIM5(SIM, Secret, application);

        claim_SIM6(SIM, Secret, vote);

        claim_SIM7(SIM, Niagree);
        claim_SIM8(SIM, Nisynch);
        claim_SIM9(SIM, Alive);
    }
}
```

Appendix B

Scyther Code (Chapter 5)

Scyther code for the security analysis of the protocol of Chapter 5.

```
usertype PublicKey;
usertype PrivateKey;
usertype Vote;
usertype Certificate;
usertype String;
usertype Application;

secret https: Function;

protocol ePayments(MER, USR, SCWS, ISS, ACQ) {

    role MER {

    const TDLS: String;
    const MIDURL: String;
    const PRMer: PrivateKey;
    const SIG: Nonce;
    const CERTMer: Certificate;
    const CERTAcq: Certificate;

    const MID: Nonce;
    const MNAME: String;
    const TPAN: Nonce;
    const TREF: Nonce;
    const sig: Nonce;
    const PUBAcq: PublicKey;
```

```
var TPAN: Nonce;
var TREF: Nonce;
var SIGpriss: Nonce;
var PRIss: PrivateKey;
var PUBMer: PublicKey;
var CERTIss: Certificate;

send_1(MER,USR, https(TDLS,MIDURL, {SIG}PRMer,CERTMer,CERTAcq));
recv_4(ISS, MER,  https({TPAN, TREF, {SIGpriss}PRIss}PUBMer, CERTIss));
send_5(MER, ACQ, {MID, MNAME, TPAN, TREF, {sig}PRMer}PUBAcq);

claim_MER1(MER, Secret, TDLS);
claim_MER2(MER, Secret, MIDURL);
claim_MER3(MER, Secret, PRMer);
claim_MER4(MER, Secret, SIG);
claim_MER5(MER, Secret, CERTMer);
claim_MER6(MER, Secret, CERTAcq);
claim_MER7(MER, Secret, MID);
claim_MER8(MER, Secret, MNAME);
claim_MER9(MER, Secret, TPAN);
claim_MER10(MER, Secret, TREF);
claim_MER11(MER, Secret, sig);
claim_MER12(MER, Secret, PUBAcq);
claim_MER13(MER, Secret, TPAN);
claim_MER14(MER, Secret, TREF);
claim_MER15(MER, Secret, SIGpriss);
claim_MER16(MER, Secret, PRIss);
claim_MER17(MER, Secret, PUBMer);
claim_MER18(MER, Secret, CERTIss);
claim_MER19(MER,Niagree);
claim_MER20(MER,Nisynch);
claim_MER21(MER,Alive);
}

role USR {

const PIN: Nonce;

var  TDLS: String;
var  MIDURL: String;
var  PRMer: PrivateKey;
var  SIG: Nonce;
var  CERTMer: Certificate;
var  CERTAcq: Certificate;

recv_1(MER,USR, https(TDLS,MIDURL, {SIG}PRMer,CERTMer,CERTAcq));
send_2( USR,SCWS, https(PIN));

claim_USR1(USR, Secret, TDLS);
claim_USR2(USR, Secret, MIDURL);
claim_USR3(USR, Secret, PRMer);
claim_USR4(USR, Secret,   SIG);
claim_USR5(USR, Secret, CERTMer);
claim_USR6(USR, Secret, CERTAcq);
claim_USR7(USR, Secret, PIN);
claim_USR8(USR,Niagree);
claim_USR9(USR,Nisynch);
claim_USR10(USR,Alive);
}

role SCWS {

const MToken: Nonce;
const SToken: Nonce;

var PIN: Nonce;

recv_2(USR, SCWS, https(PIN));
send_3(SCWS, ISS, https(MToken, SToken));

claim_SCWS1(SCWS, Secret, MToken);
claim_SCWS2(SCWS, Secret, SToken);
claim_SCWS3(SCWS, Secret, PIN);
claim_SCWS4(SCWS,Niagree);
claim_SCWS5(SCWS,Nisynch);
claim_SCWS6(SCWS,Alive);
}

role ISS {
```

```
const TPAN: Nonce;
const TREF: Nonce;
const SIG: Nonce;
const PRIss: PrivateKey;
const PUBMer: PublicKey;
const CERTIss: Certificate;

var MToken: Nonce;
var SToken: Nonce;

recv_3(SCWS, ISS, https(MToken, SToken));
send_4(ISS, MER,  https({TPAN, TREF, {SIG}PRIss}PUBMer, CERTIss));

claim_ISS1(ISS, Secret, TPAN);
claim_ISS2(ISS, Secret, TREF);
claim_ISS3(ISS, Secret, SIG);
claim_ISS4(ISS, Secret, PRIss);
claim_ISS5(ISS, Secret, PUBMer);
claim_ISS6(ISS, Secret, CERTIss);
claim_ISS7(ISS, Secret, MToken);
claim_ISS8(ISS, Secret, SToken);
claim_ISS9(ISS,Niagree);
claim_ISS10(ISS,Nisynch);
claim_ISS11(ISS,Alive);
}

role ACQ {

var MID: Nonce;
var MNAME: String;
var TPAN: Nonce;
var TREF: Nonce;
var sig: Nonce;
var PUBAcq: PublicKey;
var PRMer: PrivateKey;

recv_5(MER, ACQ, {MID, MNAME, TPAN, TREF, {sig}PRMer}PUBAcq);

claim_ACQ1(ACQ, Secret, MID);
claim_ACQ2(ACQ, Secret, MNAME);
```

```
    claim_ACQ3(ACQ, Secret, TPAN);
    claim_ACQ4(ACQ, Secret, TREF);
    claim_ACQ5(ACQ, Secret, sig);
    claim_ACQ6(ACQ, Secret, PUBAcq);
    claim_ACQ7(ACQ, Secret, PRMer);
    claim_ACQ8(ACQ,Niagree);
    claim_ACQ9(ACQ,Nisynch);
    claim_ACQ10(ACQ,Alive);
    }
}
```

Appendix C

Scyther Code (Chapter 6)

Scyther code for the security analysis of the protocol of Chapter 6.

```
usertype Password;
usertype Location;
usertype Accept;

secret https: Function;

protocol VWAuth(U, VWC, VWS, LS, RAS, SCWS) {

role U {
        const ID: Nonce;
        const pin: Password;

        var OTP: Password;

        send_1( U, VWC, ID);
        send_9( U, SCWS, https(pin));
        recv_10( SCWS, U,  https(OTP));
        send_11( U, VWC, OTP);

        claim_U1(U, Secret, pin);
        claim_U2(U, Secret, OTP);

        claim_U4(U, Niagree);
        claim_U5(U, Nisynch);
        claim_U6(U, Alive);
    }
```

```
role VWC {
        const vwc: Location;

        var ID: Nonce;
        var OTP: Password;
        var acceptance: Accept;

        recv_1( U,VWC, ID);
        send_2( VWC, VWS, https(ID));
        recv_11( U,VWC, OTP);
        send_12( VWC, VWS, https(OTP, vwc));
        recv_15( VWS, VWC, https(acceptance));

        claim_VWC1(VWC, Secret, ID);
        claim_VWC2(VWC, Secret, OTP);
        claim_VWC3(VWC, Secret, acceptance);

        claim_VWC4(VWC, Niagree);
        claim_VWC5(VWC, Nisynch);
        claim_VWC6(VWC, Alive);
    }

role VWS {

        var ID: Nonce;
        var N: Nonce;
        var phone: Location;
        var OTP: Password;
        var vwc: Location;
        var acceptance: Accept;

        recv_2( VWC, VWS, https(ID));
        send_3( VWS,LS, https(ID));
        recv_4( LS, VWS, https(N));
        send_5( VWS, RAS, https(N));
        recv_6( RAS, VWS, https(phone));
        send_7( VWS, LS, https(phone));
        recv_12( VWC, VWS, https(OTP, vwc));
        send_13( VWS,LS, https(OTP, vwc));
        recv_14( LS, VWS, https(acceptance));
        send_15( VWS, VWC, https(acceptance));

        claim_VWS1(VWS, Secret, ID);
        claim_VWS2(VWS, Secret, N);
        claim_VWS3(VWS, Secret, phone);
        claim_VWS4(VWS, Secret, OTP);
        claim_VWS5(VWS, Secret, vwc);
        claim_VWS6(VWS, Secret, acceptance);

        claim_VWS4(VWS, Niagree);
        claim_VWS5(VWS, Nisynch);
        claim_VWS6(VWS, Alive);
    }

role LS {

        const N: Nonce;
        const acceptance: Accept;

        var ID: Nonce;
        var phone: Location;
        var OTP: Password;
        var vwc: Location;

        recv_3( VWS,LS, https(ID));
        send_4( LS, VWS, https(N));
        recv_7( VWS, LS, https(phone));
        recv_13( VWS,LS, https(OTP, vwc));
        send_14(LS, VWS, https(acceptance));

        claim_ LS1( LS, Secret, ID);
        claim_ LS2( LS, Secret, N);
        claim_ LS3( LS, Secret, phone);
        claim_LS4( LS, Secret, OTP);
        claim_ LS5( LS, Secret, vwc);
        claim_ LS5( LS, Secret, acceptance);

        claim_LS6(LS, Niagree);
        claim_LS7(LS, Nisynch);
        claim_LS8(LS, Alive);
```

```
}

role RAS {

        const phone: Location;

        var N: Nonce;

        recv_5( VWS, RAS, https(N));
        send_6( RAS, VWS, https(phone));
        send_8( RAS, SCWS, https(N));

        claim_ RAS1( RAS, Secret, N);
        claim_ RAS2( RAS, Secret,phone);

        claim_RAS3(RAS, Niagree);
        claim_RAS4(RAS, Nisynch);
        claim_RAS5(RAS, Alive);
}

role SCWS {

        const OTP: Password;

        var N: Nonce;
        var pin:Password;

        recv_8( RAS, SCWS, https(N));
        recv_9( U, SCWS, https(pin));
        send_10( SCWS, U,  https(OTP));

        claim_ SCWS1( SCWS, Secret, N);
        claim_ SCWS2( SCWS, Secret, pin);
        claim_ SCWS3( SCWS, Secret, OTP);

        claim_SCWS4(SCWS, Niagree);
        claim_SCWS5(SCWS, Nisynch);
        claim_SCWS6(SCWS, Alive);
    }
}
```

Bibliography

[1] T. Berners-Lee and M. Fischetti, *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor.* HarperInformation, 2000.

[2] T. Berners-Lee, D. Dimitroyannis, A. J. Mallinckrodt, S. McKay *et al.*, "World Wide Web," *Computers in Physics*, vol. 8, no. 3, pp. 298–299, 1994.

[3] Krebs on Security. [Online]. Available: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/

[4] South Korean authorities worry about DDoS attacks ahead of elections. [Online]. Available: https://www.scmagazineuk.com/south-korean-authorities-worry-about-ddos-attacks-ahead-of-elections/article/633374

[5] Passwordless Experience. [Online]. Available: https://fidoalliance.org/download/

[6] L. Golan, A. Orad, and N. Bennett, "System and method for risk based authentication," May 5 2005, uS Patent App. 10/938,848. [Online]. Available: https://www.google.com/patents/US20050097320

[7] H. Hernandez and R. Winter, "Systems and methods for adaptive authentication," Sep. 28 2006, uS Patent App. 11/088,214. [Online]. Available: https://www.google.com/patents/US20060218393

[8] J. Bardram, R. Kjær, and M. Pedersen, "Context-aware user authentication–supporting proximity-based login in pervasive computing," in *UbiComp 2003: Ubiquitous Computing.* Springer, 2003, pp. 107–123.

[9] I. Traore, *Continuous Authentication Using Biometrics: Data, Models, and Metrics: Data, Models, and Metrics.* IGI Global, 2011.

[10] J. Wayman, A. Jain, D. Maltoni, and D. Maio, "An introduction to biometric authentication systems," *Biometric Systems*, pp. 1–20, 2005.

[11] M. Stanislav, *Two-factor authentication.* IT Governance Publishing, 2015.

[12] How much is your stolen data worth on the dark web? [Online]. Available: http://www.telegraph.co.uk/technology/internet-security/11932927/How-much-is-your-stolen-data-worth-on-the-dark-web.html

[13] J. Rees, "The challenges of PCI DSS compliance," *Computer Fraud & Security*, vol. 12 2010, no. 12, pp. 14 – 16.

[14] D. Uckelmann, M. Harrison, and F. Michahelles, *Architecting the Internet of Things*. Springer Berlin Heidelberg, 2011. [Online]. Available: https://books.google.co.uk/books?id=e1u7DAEACAAJ

[15] D. Guinard and V. Trifa, *Building the Web of Things: With Examples in Node.Js and Raspberry Pi*, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2016.

[16] "What is mirai? The malware explained." [Online]. Available: https://www.pentestpartners.com/security-blog/what-is-mirai-the-malware-explained/?doing_wp_cron=1500745894.8284900188446044921875

[17] K. E. Mayes and K. Markantonakis, Eds., *Smart Cards, Tokens, Security and Applications*. Springer, 2008.

[18] W. Rankl and W. Effing, *Smart Card Handbook*. Wiley, 2010. [Online]. Available: https://books.google.co.uk/books?id=C55-4kVUQ14C

[19] Oracle. Javacard v3.0 specifications. [Online]. Available: http://www.oracle.com/technetwork/java/javame/javacard/download/default-1492179.html

[20] ISO/IEC JTC 1/SC 17, "ISO/IEC 7816-1:2011, Identification cards – Integrated circuit cards – Part 1: Cards with contacts – Physical characteristics," 2011.

[21] R. Moreno, "Procédé et dispositif de commande électronique," *French patent FR2266222*, p. 1, 1974.

[22] ISO/IEC 7816. International Organization for Standardization. [Online]. Available: http://www.iso.org/iso/home/standards.htm

[23] ISO/IEC 14443. Identification cards – Contactless integrated circuit cards – Proximity cards . [Online]. Available: https://www.iso.org/standard/70171.html

[24] European Telecommunications Standards Institute, "ETSI TS 102 223 v14.0.0, Smart cards; Card Application Toolkit (CAT) (Release 14)," 2017.

[25] European Telecommunications Standards Institute , "ETSI TS 102 483 v8.1.0, Smart cards; UICC-Terminal interface; Internet Protocol connectivity between UICC and terminal (Release 8)," 2009.

[26] European Telecommunications Standards Institute, "ETSI TS 102 600 v10.0.0, Smart cards; UICC-Terminal interface; Characteristics of the USB interface (Release 10)," 2010.

[27] "Java Card Platform, Classic Edition 3.0.5." [Online]. Available: https://docs.oracle.com/javacard/3.0.5/index.html

[28] "Multos technical library." [Online]. Available: http://www.multos.com/developer_centre/technical_library/

[29] ISO/IEC 7816-4:2013. (2013) Identification cards – Integrated circuit cards – Part 4: Organisation, security and commands for interchange.

[30] Open EU Data Portal. [Online]. Available: https://data.europa.eu/euodp/en/data/dataset/QfZpvm83bNaQyBWPZkTA

[31] Common Criteria. [Online]. Available: https://www.commoncriteriaportal.org/cc/

[32] Federal Information Processing Standard (FIPS), "140-2," *Security Requirements for Cryptographic Modules*, vol. 25, 2001.

[33] GlobalPlatform, "Remote Application Management over HTTP Card Specification v2.2 Amendment B Version 1.1.1," March 2012.

[34] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerer's apprentice guide to fault attacks," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 370–382, 2006.

[35] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE transactions on computers*, vol. 51, no. 5, pp. 541–552, 2002.

[36] 3rd Generation Partnership Project (3GPP). Universal Subscriber Identity Module (USIM) Application Toolkit (USAT), 14.4.0, September 2017.

[37] M. Sauter, *From GSM to LTE: an introduction to mobile networks and mobile broadband.* John Wiley & Sons, 2010.

[38] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643 – 666, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128603004250

[39] M. Kenney, "Ping of death," *Insecure.org*, 1996.

[40] J. Anderson, "An analysis of fragmentation attacks," 2001.

[41] Windows 7, Vista exposed to teardrop attack. [Online]. Available: http://www.zdnet.com/article/windows-7-vista-exposed-to-teardrop-attack/

[42] "Fork bomb." [Online]. Available: http://malware.wikia.com/wiki/Fork_Bomb

[43] S. Kumar, "Smurf-based distributed denial of service (DDoS) attack amplification in Internet," in *Internet Monitoring and Protection, 2007. ICIMP 2007. Second International Conference on.* IEEE, 2007, pp. 25–25.

[44] "DDoS attacks 101: Types, targets, and motivations." [Online]. Available: https://www.calyptix.com/top-threats/ddos-attacks-101-types-targets-motivations/

[45] Imperva, "The top 10 DDoS attack trends," *White Paper*.

[46] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[47] M. B. Barcena and C. Wueest, "Insecurity in the Internet of Things," *Security Response, Symantec*, 2015.

[48] B. Partridge, "The business case for managed DDoS protection." Yankee Group, 2011.

[49] Level 3 DDoS Mitigation. [Online]. Available: http://www.level3.com/˜/media/files/brochures/en_secur_br_ddos_mitigation.pdf

[50] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart, "HTTP authentication: Basic and digest access authentication," Tech. Rep., 1999.

[51] "OpenID." [Online]. Available: http://openid.net/

[52] "OAuth." [Online]. Available: https://oauth.net/

[53] "OpenID Connect." [Online]. Available: https://oauth.net/

[54] "Security Assertion Markup Language (SAML) V2.0 Technical Overview." [Online]. Available: http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html

[55] J. Hong, "The state of phishing attacks," *Commun. ACM*, vol. 55, no. 1, pp. 74–81, Jan. 2012. [Online]. Available: http://doi.acm.org/10.1145/2063176.2063197

[56] "How was Yahoo's security breached? employee got spear phished, FBI suggests." [Online]. Available: https://arstechnica.co.uk/tech-policy/2017/03/fbi-hints-that-hack-of-semi-privileged-yahoo-employee-led-to-massive-breach/

[57] "Google Docs users hit by phishing scam." [Online]. Available: http://www.bbc.co.uk/news/business-39798022

[58] "NIST is No Longer Recommending Two-Factor Authentication Using SMS." [Online]. Available: https://www.schneier.com/blog/archives/2016/08/nist_is_no_long.html

[59] "Over 320,000 CVV codes and final records leaked online." [Online]. Available: https://hotforsecurity.bitdefender.com/blog/over-320000-cvv-codes-and-final-records-leaked-online-16645.html

[60] Payment Card Industry (PCI) Security Standards Council. (2012) Payment Card Industry Data Security Standard, Version 3.2.

[61] Smartcard-Web-Server, Approved Version 1.2.1, OMA-TS-Smartcard_Web_Server-V1_1_2_1-20130913-A, Open Mobile Alliance (OMA), Version 1.2.1, 13 Sep 2013. [Online]. Available: http://www.openmobilealliance.org

[62] Open Mobile Alliance (OMA), "Smartcard Web Server Enabler Architecture, Approved Version 1.2, 05 Mar 2013."

[63] Open Mobile Alliance (OMA), "Enabler Release Definition for Smartcard Web Server, Approved Version 1.2.1, 13 Sept 2013."

[64] Open Mobile Alliance (OMA), "Smartcard Web Server Requirements, Approved Version 1.2, 05 Mar 2013."

[65] Office for Democratic Institutions and Human Rights. (2011, March) Estonia Parliamentary Elections: OSCE/ODIHR Election Assessment Mission Final Report. [Online]. Available: http://www.osce.org/odihr/77557

[66] Geneva State Chancellery. Uncovering the veil on Geneva's Internet voting solution. Geneva Information Technology Centre. [Online]. Available: http://www.geneve.ch

[67] P. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia, "Prêt à voter: a voter-verifiable voting system," *Information Forensics and Security, IEEE Transactions on*, vol. 4, no. 4, pp. 662–673, 2009.

[68] D. Sandler, K. Derr, and D. Wallach, "VoteBox: a tamper-evident, verifiable electronic voting system," in *Proceedings of the 17th conference on Security symposium*. USENIX Association, 2008, pp. 349–364.

[69] A. Appel, M. Ginsburg, H. Hursti, B. Kernighan, C. Richards, G. Tan, and P. Venetis, "The New Jersey voting-machine lawsuit and the AVC Advantage

DRE voting machine," in *Proceedings of the 2009 conference on Electronic voting technology/workshop on trustworthy elections.* USENIX Association, 2009, pp. 5–5.

[70] T. Kohno, A. Stubblefield, A. Rubin, and D. Wallach, "Analysis of an electronic voting system," in *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on.* IEEE, 2004, pp. 27–40.

[71] Z. Xia, S. Schneider, J. Heather, P. Ryan, D. Lundin, and P. Howard, "Prêt à voter: All-in-one," 2007.

[72] B. Randell and P. Ryan, "Voting technologies and trust," *Security & Privacy, IEEE*, vol. 4, no. 5, pp. 50–56, 2006.

[73] S. Wolchok, E. Wustrow, D. Isabel, and J. Halderman, "Attacking the Washington, D.C. Internet Voting System," *16th Conference of Financial Cryptography and Data Security*, 2012.

[74] CBC/Radio Canada. NDP voting disruption deliberate, hard to track: More than 10,000 computers used in denial of service attack, voting company Scytl says. [Online]. Available: http://www.cbc.ca/news/politics/story/2012/03/27/pol-ndp-voting-disruption-deliberate.html

[75] M. Clarkson, S. Chong, and A. Myers, "Civitas: Toward a secure voting system," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on.* IEEE, 2008, pp. 354–368.

[76] B. Adida, "Helios: web-based open-audit voting," in *Proceedings of the 17th conference on Security symposium.* USENIX Association, 2008, pp. 335–348.

[77] R. Rivest, "Electronic voting," in *Financial Cryptography*, vol. 1, 2001, pp. 243–268.

[78] R. Oppliger, "How to address the secure platform problem for remote internet voting," *SIS*, vol. 2, pp. 153–173, 2002.

[79] P. Ryan and V. Teague, "Pretty Good Democracy," in *Workshop on Security Protocols*, vol. 154, 2009.

[80] J. Helbach and J. Schwenk, "Secure internet voting with code sheets," in *Proceedings of the 1st international conference on E-voting and identity.* Springer-Verlag, 2007, pp. 166–177.

[81] J. Helbach, J. Schwenk, and S. Schage, "Code voting with linkable group signatures," in *EVOTE08, 3rd International Workshop on Electronic Voting*, 2008.

[82] SANS Password Policy. [Online]. Available: http://www.sans.org/security-resources/policies/

[83] Recommendation for Key Management - Part 1: General. National Institute of Standards and Technology (NIST) Special Publication 800-57, Part 1, Revision 4, January 2016. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-57/

[84] D. Chaum, P. Ryan, and S. Schneider, "A practical voter-verifiable election scheme," *Computer Security–ESORICS 2005*, pp. 118–139, 2005.

[85] P. Ryan and S. Schneider, "Prêt à voter with re-encryption mixes," *Computer Security–ESORICS 2006*, pp. 313–326, 2006.

[86] P. Ryan, "Prêt à voter with confirmation codes," in *Proceedings of the USENIX Electronic Voting Technology Workshop*, 2011.

[87] L. Zhou, F. Schneider, M. Marsh, and A. Redz, "Distributed blinding for distributed ElGamal re-encryption," in *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on.* IEEE, 2005, pp. 824–824.

[88] OWASP Top Ten Project. [Online]. Available: https://www.owasp.org

[89] Department of Computer Science, Oxford University, "The scyther tool." [Online]. Available: https://www.cs.ox.ac.uk/people/cas.cremers/scyther/

[90] C. J. Cremers, "The scyther tool: Verification, falsification, and analysis of security protocols," in *CAV*, vol. 8. Springer, 2008, pp. 414–418.

[91] C. J. Cremers, "Unbounded verification, falsification, and characterization of security protocols by pattern refinement," in *Proceedings of the 15th ACM conference on Computer and communications security.* ACM, 2008, pp. 119–128.

[92] C. J. Cremers, P. Lafourcade, and P. Nadeau, "Comparing state spaces in automatic security protocol analysis," in *Formal to Practical Security.* Springer, 2009, pp. 70–94.

[93] D. Basin, C. Cremers, and S. Meier, "Provably repairing the ISO/IEC 9798 standard for entity authentication 1," *Journal of Computer Security*, vol. 21, no. 6, pp. 817–846, 2013.

[94] C. Cremers and M. Horvat, "Improving the ISO/IEC 11770 standard for key management techniques," *International Journal of Information Security*, vol. 15, no. 6, pp. 659–673, Nov 2016. [Online]. Available: https://doi.org/10.1007/s10207-015-0306-9

[95] VIIIth Revisionary Greek Parliament, “The Constitution of Greece, Article 5A2,” 2008.

[96] Ministry of Transport and Communications of Finland, “1 Mbit Internet access a universal service in Finland from the beginning of July,” 2009. [Online]. Available: http://www.lvm.fi/web/en/pressreleases/-/view/1169259

[97] Forrester Research. (2012, February) Forrester Research Online Retail Forecast, 2011 To 2016 (Western Europe).

[98] Sachin Shetty, Symantec, “Introduction to Spyware Keyloggers,” November 2010, http://www.symantec.com/connect/articles/introduction-spyware-keyloggers.

[99] The New York Times, “MasterCard and Visa Investigate Data Breach,” 30 March 2012. [Online]. Available: http://www.nytimes.com/2012/03/31/business/mastercard-and-visa-look-into-possible-attack.html

[100] Linda McGlasson, “Bank of New York Mellon Investigated for Lost Data Tape,” 27 May 2008. [Online]. Available: http://www.bankinfosecurity.com/bank-new-york-mellon-investigated-for-lost-data-tape-a-862/op-1

[101] Payment Card Industry (PCI) Security Standards Council, “Why Comply with PCI Security Standards.” [Online]. Available: https://www.pcisecuritystandards.org/security_standards/why_comply.php

[102] Kaspersky Labs, “Protecting your bank account using Safe Money technology,” Tech. Rep.

[103] Trusteer, “Trusteer Rapport for Online Banking.” [Online]. Available: http://www.trusteer.com/Products/Trusteer-Rapport-for-Online-Banking

[104] JPMorgan, “2012 Online Fraud Report,” Tech. Rep., 2012.

[105] Citigroup Inc., “Virtual Account Numbers.” [Online]. Available: https://www.citibank.com/us/cards/vanpromo/cmc_pop/index2.htm

[106] MasterCard, “Frequently Asked Questions, The MasterCard Site Data Protection (SDP) Program.” [Online]. Available: https://www.mastercard.com/us/company/en/docs/Frequently%20Asked%20Questions%20_March%2022%202012.pdf

[107] RFC 2818, “Hypertext Transfer Protocol over TLS protocol, May 2000, http://www.ietf.org/rfc/rfc2818.txt.”

[108] Amazon.com, Inc. [Online]. Available: http://www.amazon.com

[109] PayPal Inc. [Online]. Available: https://www.paypal.com

[110] Google Inc. [Online]. Available: http://www.google.com/wallet/

[111] Netregistry, "What is a payment gateway." [Online]. Available: http://www.netregistry.com.au/resources/what-is/ecommerce/what-is-a-payment-gateway/

[112] ISO/IEC 18092:2004, Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1).

[113] ISO/IEC 21481:2012, Information technology – Telecommunications and information exchange between systems – Near Field Communication Interface and Protocol -2 (NFCIP-2).

[114] Mohsen Toorani, Ali A. Beheshti, "SSMS - A Secure SMS Messaging Protocol for the M-Payment Systems," in *Proceedings of the 13th IEEE Symposium on Computers and Communications (ISCC'08), pp.700-705*, 2008.

[115] USA Today, "Pay by phone: More merchants embrace direct mobile billing," 4 April 2012. [Online]. Available: http://usatoday30.usatoday.com/tech/news/story/2012-04-04/mobile-billing-boku-zong/54003414/1

[116] RFC 5280, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, May 2008, http://tools.ietf.org/html/rfc5280."

[117] D. Goodin, "Crack in Internet's foundation of trust allows HTTPs session hijacking," September 2012, http://arstechnica.com/security/2012/09/crime-hijacks-https-sessions/.

[118] National Institute of Standards and Technology (NIST), "NIST Special Publication 800-57, Recommendation for Key Management Part 1: General (Revision 3)," July 2012.

[119] K. Eagles, C. Markantonakis, and K. Mayes, "A comparative analysis of common threats, vulnerabilities, attacks and countermeasures within smart card and wireless sensor network node technologies," in *WISTP*, 2007, pp. 161–174.

[120] P. Syverson, "A taxonomy of replay attacks," in *Proceedings of the 7th IEEE Computer Security Foundations Workshop.* Society Press, 1994, pp. 187–191.

[121] M. Bell, "Virtual Worlds Research: Past, Present & Future July 2008: Toward a definition of "Virtual Worlds"," 2008.

[122] KZero. (2012, March) Virtual worlds: Industry and user data universe. [Online]. Available: www.kzero.co.uk

[123] Second Life Official Website. http://www.secondlife.com/. [Online]. Available: http://secondlife.com/

[124] World of Warcraft. http://eu.battle.net/wow/en/. [Online]. Available: http://eu.battle.net/wow/en//

[125] G. McGraw and M. Chow, "Guest editors' introduction: Securing online games: Safeguarding the future of software security: How world of warcraft almost ruined my credit rating," *Security & Privacy, IEEE*, pp. pp. 11–12, 2009. [Online]. Available: http://www2.computer.org/cms/Computer.org/dl/mags/sp/2009/03/extras/msp2009030011s.pdf

[126] J. Yardley. (2013) Unter den linden, picture made at the 1920s berlin project in second life. Licensed under; https://creativecommons.org/licenses/by-sa/3.0/deed.en. [Online]. Available: https://commons.wikimedia.org/wiki/File:Unter_den_Linden,_picture_made_at_the_1920s_Berlin_Project_in_Second_Life.png

[127] I. Jorstad, T. Jonvik *et al.*, "Strong authentication with mobile phone as security token," in *Mobile Adhoc and Sensor Systems, 2009. MASS'09.* IEEE, 2009, pp. 777–782.

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

[129] Juniper Networks Inc. 2011 Mobile Threats Report. Juniper Networks Inc.

[130] Neustar, "IP Geolocation: A Valuable Weapon to Fight Online Card-Not-Present Payment Fraud," http://www.neustar.biz/enterprise/docs/whitepapers/ip-intelligence/geolocation-detecting-card-not-present-fraud.pdf.

[131] Calvin, J. and Dickens, A. and Gaines, B. and Metzger, P. and Miller, D. and Owen, D., "The SIMNET virtual world architecture," in *Virtual Reality Annual International Symposium, IEEE*, September 1993, pp. pp. 450–455.

[132] Frecon, Emmanuel, and Marten Stenius, "DIVE: A scaleable network architecture for distributed virtual environments," in *Distributed Systems Engineering 5.3*, 1998.

[133] Fernandes, S. and Antonello, R. and Moreira, J. and Kamienski, C. and Sadok, D., "Traffic analysis beyond this world: the case of Second Life," in *17th International workshop on Network and operating systems support for digital audio and video, University of Illinois, Urbana-Champaign*, 2007.

[134] RFC 4226. HOTP: An HMAC-Based One-Time Password Algorithm, December 2005. http://www.ietf.org/rfc/rfc4226.txt.

[135] SANS Institute InfoSec Reading Room, "Introduction to IP Spoofing," November 2000.

[136] P. Gill, Y. Ganjali, B. Wong, and D. Lie, "Dude, where's that IP? Circumventing measurement-based IP geolocation," in *Proceedings of the 19th USENIX conference on Security*, Berkeley, CA, USA, 2010.

[137] K. Ashton. (2009, June) That 'Internet of Things' Thing. [Online]. Available: http://www.rfidjournal.com/articles/view?4986

[138] Gartner. The Internet of Things Enables Digital Business. [Online]. Available: http://www.gartner.com/technology/research/internet-of-things/

[139] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[140] S. Duquennoy, G. Grimaud, and J.-J. Vandewalle, "The Web of Things: interconnecting devices with high usability and performance," in *Embedded Software and Systems, 2009. ICESS'09. International Conference on.* IEEE, 2009, pp. 323–330.

[141] D. Guinard and V. Trifa, "Towards the Web of Things: Web mashups for embedded devices," in *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain*, 2009, p. 15.

[142] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the Web of Things," in *Internet of Things (IOT), 2010.* IEEE, 2010, pp. 1–8.

[143] D. Zeng, S. Guo, and Z. Cheng, "The Web of Things: A survey," *Journal of Communications*, vol. 6, no. 6, pp. 424–438, 2011.

[144] D. Guinard, V. M. Trifa, and E. Wilde, *Architecting a mashable open World Wide Web of Things.* ETH, Department of Computer Science, 2010.

[145] V. Trifa, S. Wieland, D. Guinard, and T. M. Bohnert, "Design and implementation of a gateway for web-based interaction and management of embedded devices," *Submitted to DCOSS*, 2009.

[146] M. Kovatsch, M. Weiss, and D. Guinard, "Embedding internet technology for home automation," in *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on.* IEEE, 2010, pp. 1–8.

[147] B. Ostermaier, F. Schlup, and K. Romer, "Webplug: A framework for the Web of Things," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on.* IEEE, 2010, pp. 690–695.

[148] Web of Things Framework. [Online]. Available: https://github.com/w3c/web-of-things-framework

[149] T. Dierks, "The Transport Layer Security (TLS) protocol version 1.2," 2008.

[150] Facebook. (2008, May) Announcing Facebook Connect. [Online]. Available: https://developers.facebook.com/blog/post/2008/05/09/announcing-facebook-connect/

[151] Kerberos: The Network Authentication Protocol. [Online]. Available: http://web.mit.edu/kerberos/

[152] Microsoft. Microsoft Kerberos. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa378747%28v=vs.85%29.aspx

[153] Tracking the FREAK Attack. [Online]. Available: https://freakattack.com/

[154] W. Chen, K. Mayes, Y. Lien, and J. Chiu, "NFC mobile payment with Citizen Digital Certificate," in *Next Generation Information Technology (ICNIT), 2011 The 2nd International Conference on.* IEEE, 2011, pp. 120–126.