Centralized Vehicle License Management on ERTMS

Key Database and Communication between Server and RBC

EVANGELOS KROKOS
Centralized Vehicle License Management on ERTMS
Key Database and Communication between Server and RBC

Master’s Thesis
Stockholm, October 16, 2016
Abstract

Today’s rail industry is under great demand. People will use trains even more in the future. Rail industry has to develop and integrate new systems towards a secure transportation system. One of these steps is investigated in this master thesis. Until now, many sensitive and vital data were transferred by using old methods like USB storage devices. In this master thesis, a new and secure way to store authentication keys in a central database is described as well as a secure way of communication between this database and the RBC entities. Different technologies are used and have as result the proposed goal. The creation of a database with PHP and the implementation of a server-client communication using TLS and HTTPS are described in this project. Improvements and future development of the project are discussed at the end.

Keywords
TLS, HTTPS, RBC, KMAC, KTRANS, ERTMS
Abstract


Nyckelord
TLS, HTTPS, RBC, KMAC, KTRANS, ERTMS
To my beloved grandfather and parents.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>4</td>
</tr>
<tr>
<td>Abstract</td>
<td>5</td>
</tr>
<tr>
<td>Contents</td>
<td>8</td>
</tr>
<tr>
<td>List of Figures</td>
<td>9</td>
</tr>
<tr>
<td>List of Tables</td>
<td>9</td>
</tr>
<tr>
<td>Abbreviations and Acronyms</td>
<td>10</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>12</td>
</tr>
<tr>
<td>1.1 Preface</td>
<td>12</td>
</tr>
<tr>
<td>1.2 Problem</td>
<td>12</td>
</tr>
<tr>
<td>1.3 Goal</td>
<td>12</td>
</tr>
<tr>
<td>1.4 Conditions &amp; Limitations</td>
<td>12</td>
</tr>
<tr>
<td>2 Background</td>
<td>14</td>
</tr>
<tr>
<td>2.1 Rail Industry</td>
<td>14</td>
</tr>
<tr>
<td>2.2 Bombardier Transportation</td>
<td>15</td>
</tr>
<tr>
<td>2.3 ERTMS</td>
<td>16</td>
</tr>
<tr>
<td>2.3.1 Application Level 0</td>
<td>17</td>
</tr>
<tr>
<td>2.3.2 Application Level STM / NTC</td>
<td>18</td>
</tr>
<tr>
<td>2.3.3 Application Level 1</td>
<td>18</td>
</tr>
<tr>
<td>2.3.4 Application Level 2</td>
<td>19</td>
</tr>
<tr>
<td>2.3.5 Application Level 3</td>
<td>20</td>
</tr>
<tr>
<td>2.4 GSM-R</td>
<td>21</td>
</tr>
<tr>
<td>2.5 ETCS</td>
<td>22</td>
</tr>
<tr>
<td>3 Current Procedure</td>
<td>24</td>
</tr>
<tr>
<td>4 The Internet</td>
<td>27</td>
</tr>
<tr>
<td>4.1 Description</td>
<td>27</td>
</tr>
<tr>
<td>4.2 TLS</td>
<td>28</td>
</tr>
<tr>
<td>5 Proposed Solution</td>
<td>30</td>
</tr>
<tr>
<td>5.1 Database</td>
<td>31</td>
</tr>
<tr>
<td>5.2 Apache server &amp; Apache tomcat</td>
<td>32</td>
</tr>
<tr>
<td>5.3 Servlets</td>
<td>33</td>
</tr>
<tr>
<td>5.4 Certificates and Certification Authority</td>
<td>34</td>
</tr>
<tr>
<td>5.5 DNS</td>
<td>36</td>
</tr>
<tr>
<td>5.6 Firewalls &amp; IDS</td>
<td>36</td>
</tr>
<tr>
<td>5.7 Test</td>
<td>37</td>
</tr>
<tr>
<td>6 Results and Future</td>
<td>38</td>
</tr>
<tr>
<td>6.1 Related Work</td>
<td>38</td>
</tr>
<tr>
<td>6.2 Sustainability</td>
<td>39</td>
</tr>
<tr>
<td>A APPENDIX 1</td>
<td>40</td>
</tr>
</tbody>
</table>
List of Figures

1  ECTS Overview - Inspired by [15]  ........................................... 17
2  Application Level 0 - Inspired by [7]  ........................................ 18
3  Application Level STM / NTC - Inspired by [7]  .......................... 18
4  Application Level 1 - Inspired by [7]  ........................................ 19
5  Application Level 2 - Inspired by [7]  ........................................ 20
6  Application Level 3 - Inspired by [7]  ........................................ 20
7  GSM-R - Inspired by [20]  ...................................................... 22
8  Internet Models - Inspired by [35] - Created with [76]  .................. 28
9  Internet Models - Inspired by [38] - Created with [76]  .................. 29
10 Overview of the implentation  .................................................... 30
11 Proxy and Web Server - Inspired by [58] - Created with [76]  ....... 33
12 Servlet example - Inspired by [61] - Created with [76]  ................. 34
13 Number of requests - Created with [67]  .................................... 37
14 Time in (ms) of replying in a request - Created with [67]  .......... 37
15 First part of capture - Created with [75]  ................................. 40
16 Second part of capture - Created with [75]  ............................... 40
17 Third part of capture - Created with [75]  ................................. 41
18 Fourth part of capture - Created with [75]  ............................... 41

List of Tables

1  Spread of Railways in Europe during the 19th century (km) -
   Inspired by [3]  ................................................................. 14
2  Offline keys hierarchy - Inspired by [29]  ................................. 25
## Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>Advanced Encryption Standard</td>
</tr>
<tr>
<td>CA</td>
<td>Certificate Authority</td>
</tr>
<tr>
<td>CBC</td>
<td>Cipher Block Chaining</td>
</tr>
<tr>
<td>CN</td>
<td>Canonical Name</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Process Unit</td>
</tr>
<tr>
<td>DCS</td>
<td>Digital Cellular System</td>
</tr>
<tr>
<td>DES</td>
<td>Data Encryption Standard</td>
</tr>
<tr>
<td>DMI</td>
<td>Driver Machine Interface</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>E-GSM</td>
<td>Extended Global System for Mobile Communications</td>
</tr>
<tr>
<td>ERTMS</td>
<td>European Rail Traffic Management System</td>
</tr>
<tr>
<td>ETCS</td>
<td>European Train Control System</td>
</tr>
<tr>
<td>GSM-R</td>
<td>Global System for Mobile Communications - Railway</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>HTTP Secure</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>KDC</td>
<td>Key Distribution Centre</td>
</tr>
<tr>
<td>KMAC</td>
<td>Authentication Key</td>
</tr>
<tr>
<td>KMC</td>
<td>Key Management Centre</td>
</tr>
<tr>
<td>KMS</td>
<td>Key Management System</td>
</tr>
<tr>
<td>KTRANS</td>
<td>Transport Key</td>
</tr>
<tr>
<td>LEU</td>
<td>Lineside Electronics Unit</td>
</tr>
<tr>
<td>MAC</td>
<td>Message Authentication Code</td>
</tr>
<tr>
<td>NTC</td>
<td>National Train Control</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>RBC</td>
<td>Radio Block Centre</td>
</tr>
<tr>
<td>RIU</td>
<td>Radio Infill Unit</td>
</tr>
<tr>
<td>SHA</td>
<td>Secure Hash Algorithm</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>3-DES</td>
<td>Triple-Data Encryption Standard</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Preface
This master thesis project describes the current way to store authentication keys in railway industry and their current way to manage them. The project aims to come up with a secure way to store and manage these keys. The thesis is carried out at Bombardier Transportation in Stockholm.

1.2 Problem
The thesis work should be a feasibility study on introducing a Centralized Vehicle Licence / key-handling. The scope of the thesis is to develop a proposal for the management of vehicle keys that are centralized and can be updated without a traffic impact, instead of having key management statically programmed in the wayside equipment as it is today. When the vehicles connect, "calling" wayside equipment shall not have a local list, but instead make a call to the central "Vehicle Key database", which then responds if the vehicle is ok or not. The availability of the infrastructure should be a key point for the system.

To operate on an ERTMS route, each individual vehicle must have an encrypted key installed. When the vehicle is ready to go out on the route, the vehicle key is confirmed by the wayside systems that controls the keys and sees if the vehicle is approved for entry on the track. The process to update the list of vehicles in the soil system is involving both extensive administrative operations as well as updating the software in the wayside equipment (it is manually installed in both on-board unit and RBCs). When updating the list of vehicles, the wayside system needs a restart which means that all operation on the route/line needs to be stopped.

1.3 Goal
The scope of the project will be a centralized management of the authentication keys used in rail industry. There is neither a standardized method to handle the keys in a central way nor a secure way to store them. Moreover, the novelty should be on this aspect: how and what security mechanisms should be used to have a secure communication using these authentication keys among the different entities of ERTMS/ETCS specifications and more precisely, how the RBCs will communicate with the KMC. There should be a detailed description of the crypto algorithms and all the tools for the implementation.

1.4 Conditions & Limitations
In On-line Key Management FFFIS (Subset-137) [1], which was published by the Union of Signaling Industry (UNISIG), the future development of ERTMS is described. In this subset, among other specifications, the management of the on-line distribution of cryptographic keys between KMCs is being outlined, as well as the distribution among KMCs and other parts of the system, such as RBCs and On-board equipment. In the on-line interface overview chapter, there is an analysis of the security mechanisms that will be used. They describe what a KMC should achieve and how. In the project, a test KMC which will use TLS
[2] is being created, but its limitation is that it will not have the full properties of a KMC as it is described in the Subset-137. Moreover, the authentication keys used (KMAC) are for testing purposes only and we suppose that already someone else created them.
2 Background

2.1 Rail Industry

The existence of railway in Europe holds its roots at the beginning of 19th century. Back in these years, many European countries started developing their own legacy systems. The very first line was established in England between Stockton and Darlington and was about 39 km long [3]. As the economy and societies kept growing, the continent had to deal with the enormous demand for shipment of goods and transportation of people. As a result, there was a booming period for the expansion of rail network. The numbers are showing the range of this extensive construction rate, which created a great network of railways (Table 1) [4].

<table>
<thead>
<tr>
<th>Country</th>
<th>1840</th>
<th>1860</th>
<th>1880</th>
<th>1900</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austria - Hungary</td>
<td>144</td>
<td>4.543</td>
<td>18.507</td>
<td>36.330</td>
</tr>
<tr>
<td>Belgium</td>
<td>334</td>
<td>1.730</td>
<td>4.112</td>
<td>4.591</td>
</tr>
<tr>
<td>France</td>
<td>496</td>
<td>9.167</td>
<td>23.089</td>
<td>38.109</td>
</tr>
<tr>
<td>Germany</td>
<td>469</td>
<td>11.089</td>
<td>33.838</td>
<td>51.678</td>
</tr>
<tr>
<td>Great Britain</td>
<td>2.390</td>
<td>14.603</td>
<td>25.060</td>
<td>30.079</td>
</tr>
<tr>
<td>Italy</td>
<td>20</td>
<td>2.404</td>
<td>9.290</td>
<td>16.429</td>
</tr>
<tr>
<td>Netherlands</td>
<td>17</td>
<td>335</td>
<td>1.846</td>
<td>2.776</td>
</tr>
<tr>
<td>Spain</td>
<td>-</td>
<td>1.917</td>
<td>7.490</td>
<td>13.214</td>
</tr>
</tbody>
</table>

Table 1: Spread of Railways in Europe during the 19th century (km) - Inspired by [3]

The development continued intensively during the next century. During the two World Wars, the railway network dealt with extensive damages. The following years, large-scale projects expanded the existing railways and created new, which also enabled the connection between the European countries.

Coming to nowadays, many changes have been made to the railway industry. Most of the railways enable the trains to run with electricity, the communication systems are developed even more and include technologies from the computer science field. More than ever, trains need to be as secure as a bank or another really crucial system of our society.

During the last decades, computer science and its components brought enormous changes in our lives. Almost everything nowadays bears a printed circuit board (PCB) [5] and there is extensive use of memory and Central Process Unit (CPU) [6] while in comparison to the previous years, those components could only be seen in high expensive computer systems. Except from this, the tremendous explosion of the Internet brought the need for connection and communication between different systems. As a result the field of rail industry had to follow the tense and include the new technology, which will bring rail industry in the new century.

European Union in the contents of European integration started to develop a new system, which was going to integrate all the different rail systems around
the Union and moreover Schengen’s area. All those years, each country in Europe developed its own rail system, which had many differences compared to systems in the neighboring countries. The diversity was in many senses, from the width of the rails, the electricity supply, the signaling systems, the wayside equipment used etc. Therefore, European Commission towards a new era for railways in Europe, started a project for an integrated European railway system. The investigation of this system started back in 1989 [7], when the first discussion about rail signaling and train control started by the European Transport minister. Following this, the next steps established the first specifications for the on-board equipment, as well as the wayside and track equipment (EUROCAB [7], EUROBALISES [8], and EURORADIO [8]). Later, most rail companies established a consortium (UNISIG) [9] under European’s Union demands and created the final specifications for the system known as European Rail Traffic Management System (ERTMS) [10]. One of these companies was Bombardier Transportation. Bombardier Transportation’s background and products are described in the next chapter.

2.2 Bombardier Transportation

Bombardier is one of the biggest companies in rail industry. It started its services back in 1930 as a snowmobile producer in Canada. During the next decades the portfolio of the company expanded and started going into the production of aircrafts. Some very famous products of Bombardier are the firefighting CL-415 and the new coming C-Series jet. Around 1970, Bombardier is starting its rail part by acquiring Lohnerwerke, an Austrian manufacturer of motor scooters and trams. Continuing its business in aerospace and transportation the next decades, Bombardier nowadays reaps the benefits of all these years by being one of the top companies in aerospace and the leading company in rail business. [11]

Today, Bombardier employs around 71000 people in both sectors. Almost half of the employees of the company belong to the transportation part. Sweden is playing an important role in the chain of products for Bombardier. Some departments with different roles exist in many cities. The global headquarters of the Rail Control Solutions (RCS) department of Bombardier’s Transportation, which are liable for control and system division, are located in Stockholm. The department includes both research and production stages. More than 3500 people are working in this division around the world. In Stockholm, around 450 people are responsible for optimising the flow of the trains, developing and customizing the track-side and on-board systems, and developing ERTMS. In addition, supplementary projects are also established, like sales, project management, logistics, sales and logistics. RCS in Stockholm is responsible for the north region in which belong the Nordic countries, Baltic countries, Russia, OSS countries, Turkey, South Africa and Slovakia. [12]

Bombardier Transportation and its department Rail Control Solutions (RCS) offer a diversity of services. Among them there are Engineering and product development, Safety assurance, Project management and Services. As for ERTMS, RCS department has a strong involvement with some highlights, like the highest speed lines in China and the first ERTMS lines in Sweden and Europe’s busiest line.

Until now, many lines around Europe and globally still employ under Level 1 or STM systems. Towards the development of new generation of ERTMS
Level 2 and moreover Level 3, many different systems have to change. The effort for this is great since there are thousands kilometers of lines and wayside systems that have to change in order to fulfill the requirements for Level 2 and 3. RCS department is currently working on this considerable and extensive project. Part of this project is the current master thesis and its objective is to present a new way to store/edit new RBC [13] and Train KMAC [14] keys in a central station, as well as develop a new way to secure manage and deliver them to the system.

2.3 ERTMS

European Traffic Control System (ETCS) Figure 1 [15] is the hardware and software which needs to be developed in order to put the concepts of European Rail Traffic Management System (ERTMS) into operation. The great reason that European Union is moving towards a universal system is the need for a trans-European universal system that will allow the interoperability between different systems and technologies from different countries. The previous situation was a plethora of different national systems in each country which was making difficult for a train to travel through the countries of the continent. These systems, which are named in the ERTMS specification as Specific Transmission Module (STM), had some of the capabilities and properties of the new standardized levels of ERTMS, but not in the same uniformly way.

ERTMS is designed to solve this situation and become a standard that should exist in every country of Europe. The standard describes two Classes and four application Levels of traffic control systems, which are known as Level 0-3. The two Classes are Class A and Class B. First, the Class A includes the Control, Command and Signaling functions, interfaces and performances that are described in ERTMS specifications. Class B includes the already existing control command and signaling functions, interfaces and performances in the different countries. Application Levels will be described in the next sections.
2.3.1 Application Level 0

In this level, train is equipped with ERTMS/ETCS equipment and the line is also equipped with ERTMS/ETCS infrastructure. Level 0 has limited capabilities and the only on-board equipment used is the supervision of the maximum speed. There is no information regarding authorization passing from the track-side. The maximum speed restriction is being passed to driver through a balise. Driver observes the roadside signals, but these signals may be different in different levels (Figure 2).
2.3.2 Application Level STM / NTC

Here is the STM / NTC level. It was the system that was employed in every country before the arrival of ERTMS specifications. Trains with ERTMS/ETCS equipment, which travel on STM lines, need to be equipped with STM’s equipment as well. Supervision and functionality of every system varies among the different systems (Figure 3).

2.3.3 Application Level 1

Level 1 is designed as an overlay to an existing trackside infrastructure existing from interlocking, lineside signals and track vacancy [16]. Level one is a
spot transmission system, where continuous supervision of the train speed and location occurs. The Movement Authority determined for this purpose will be transmitted to the ETCS on board equipment via Eurobalises and optionally by Euroloop or Radio Infill Units. Movement authority is passed to trains by controlled Eurobalises, which receive these data from trackside infrastructure such as Lineside Electronics Unit (LEU) [17] and then beacon them to trains. In addition, fixed Eurobalises pass positioning data to trains. Train detection on the line is done by track circuits or axle counters. On board equipment continuously monitors and calculates the maximum permitted speed and the braking curve. Euroloop can also pass moving authorization on trains but in a continually manner, over cables emitting electromagnetic waves [18].

![Application Level 1 - Inspired by [7]](image)

2.3.4 Application Level 2

It is also designed as an overlay system to an existing infrastructure. Train detection and train integrity supervision is done by the wayside entities. In this Level, the Radio Block Centre (RBC) [13] is introduced. In addition, for bi-directional communication between the ETCS Train borne unit and the RBC, GSM-R [19] is being used. GSM-R is the railway edition of 2G of telecommunications GSM. It continuously passes movement authority to the on board unit through RBC’s. On the other hand, trains send their position through GSM-R at regular intervals based on the information obtained from the fixed Eurobalises. Track detection and train integrity are done by the trackside as in the previous levels. Even in this Level, we still need many wayside equipments to have a train safe on the rails [18].
2.3.5 Application Level 3

Level 3 is the most sophisticated and the closest Level towards the full automation of the signaling system. Trains will receive authorization to run from the RBC using GSM-R. No lineside signals or trackside detection are needed and driver will receive all necessary information by looking at the ETCS DMI. Train movements are supervised according to the information received from the RBC. The train sends its data about speed and current position through GSM-R. Balises are giving fixed messages and identify the train’s position. Train integrity is achieved on the train and is passed to RBCs [18].

ERTMS consists of two different components:
− Global System for Mobile Communications à“ Railway (GSM-R), a protocol created for use only in railway systems. The first task of the protocol was the communication between the driver and the employees in the stations.

− European Train Control System (ETCS), as the name implies, is the standard that was created from European Union and is mandatory for the implementation of ERTMS specifications.

2.4 GSM-R

Global System for Mobile Communications is the dedicated version of the plain GSM, from the telecommunications field, for use only in railway industry. It is based in the requirements and specifications from the European Integrated Radio Enhanced Network (EIRENE) [20] and Mobile radio for Railway Networks in Europe (MORANE) [21] projects. The main difference from the simple GSM is the frequency band. GSM-R is working in either the E-GSM (900 MHz-GSM) or DCS 1800 (1,800 MHz-GSM). In Europe, 900MHz and more specific with a frequency range of 806-960 MHz are mainly used. In general, the 876-915 MHz is used for upload and 921-960 MHz for download. In Europe, the 876 MHz to 880 MHz is used for download and the 921 MHz to 925 MHz bands are used for data download. Channel spacing is 200 kHz [5], [6].

As is specified in the requirements for GSM-R, there are some features specialized for this wireless protocol:

− fast call set-up for railway emergency calls
− priority and pre-emption
− group-calls
− voice broadcast calls
− railway emergency calls
− functional numbering
− location dependent addressing

GPRS is also used in GSM-R and is responsible for transferring data.
Figure 7: GSM-R - Inspired by [20]

2.5 ETCS

The European Train Control System (ETCS) is on its way to be the standard railway system around Europe. In its full deployment, it will be the basic standard of communication for both trains (on-board systems) and wayside equipment. As train density is growing, line signals will be demolished. This is the area that ETCS is acting together with the enhancing interoperability in cross border lines.

The ERTMS/ETCS specifications [22] describe all the entities that should be part of the whole system. A full implementation of the system includes the train, the driver of the train, some interfaces on the train and finally some external trackside systems. ERTMS has different levels that can be applied. But here we can describe all the subsystems that can be involved. These subsystems are:

1. Balise
2. Lineside electronic unit
3. Radio communication network (GSM-R)
4. Radio Block Centre (RBC)
5. Euroloop
6. Radio infill unit
7. Key Management Centre (KMC) [23]
8. Public Key Infrastructure (PKI) [24]
Balise or Eurobalise as is called in the ERTMS specifications, is a component sitting in the middle of the tracks and communicating with the on-board systems on the trains. They are sending telegrams to the on-board systems, which can be fixed or not. Their duty is to validate the position of the train and its direction. In most of the cases they are ordered in groups.

Lineside electronic unit is sending information to the fixed balises, which then they send telegrams to the on-board equipment. The information is coming from other equipment like Interlocking, RBCs and also Euroloops and Radio infill units.

Radio communication network (GSM-R) is described above.

Radio Block Centre (RBC) is used in Levels 2 and 3. It is basically a computer system (server), which computes the messages that should be send to the on-board systems on the trains. They ensure the safety of the trains and giving movement authorities to them. They are in close collaboration with the trackside units, such as balise and with on-board units as well. The messages are exchanged with GSM-R. They handle a big area and at the end of it, they should handover the train to the next RBC.

Euroloop is a leaky cable that can send continuously information to the on-board system. It is used only in the Level 1 of the system.

Radio infill unit as is described in [25] is "providing signaling information in advance as regard to the next main signal in the train running direction".

Key Management Centre (KMC) is handling and managing the cryptographic keys, which are used for communication among the different systems. These keys, as we will see in the next section, are the KTRANS and KMAC. The role of the KMC is to generate, share, delete and update these keys.

Public Key Infrastructure (PKI) is the infrastructure chosen by the UNISIG consortium to establish secure communications among the ERTMS/ETCS units for transferring their authentication keys. All the specifications are described in the Subset-137 [1].
3 Current Procedure

The current working procedure of the rail system in Europe can be seen in the following steps. When a train is out and ready to start its journey, countless of systems and subsystems are engaged to make this happen. In this master thesis the reference of a train that is ready for a trip, will mean a train running from Level 2 and above, since this is the first Level that RBC is engaged and can be used.

Accordingly, in a Level 2 or 3 ERTMS system, when a train is travelling on a railway, this means that it is authenticated to run there. This authentication is being done from the moment a train is in the responsible area of a RBC. RBCs are in general responsible for an area of 100 kilometers and therefore in each country there are some hundreds or thousands, depending on load and the traffic of their area of course. The authentication is possible, since each train and each RBC bear a KMAC, which is the key created by Trafiverket in Sweden, and then these keys are stored manually to every RBC and train. This key is only a part of a relevant long procedure of authentication, which is described later in the thesis.

Nowadays, not many railways can handle Level 2 or 3. There are many reasons for this, with the most common being the cost to change the system. Following the change of the level, countless subsystems will need to be changed, therefore many railways still working with Level 1 or their previous national systems (STM) will have to be changed as well. Moreover, all these lines are used from different companies with different systems. So there is another reason why ERTMS/ETCS is being established, to lower the cost of development in the companies since they will have the same standards. The focus of this thesis, will be the creation of a secure central way (KMC) to store KMACs, as well as a secure way of communication between the RBC and the KMC.

Having said that, we should explain how the communication between an RBC and an on-board system is being done at the moment. When a train is running in the area of a RBC in Level 2 and 3 these two systems should authenticate each other. This authentication is mandatory since the train is taking now the moving authority and all the related data from the RBC, therefore we need to clarify who is talking with whom. In other words, a train should be able to run where it is authenticated to run. The same should be for the RBC as well. An RBC might not include all the trains that exist in the country.

When a train runs on a Level 2 or 3 in ERTMS specifications, all the authentication keys are stored manually at the on board (train) and wayside (RBC) equipment. A trusted person is used from Trafikverket [26], Sweden’s national transport agency. This person is responsible to bring the keys (KTRANS and KMACs) with a USB flash drive or a CD-ROM and store them locally in on board and RBC systems as well.

As someone can imagine there are many vulnerabilities and security issues in this procedure. For example, someone can steal the keys by just having access to this USB drive, or easier the flash driver could be lost by an authenticated person and an adversary could find it. Even without the specific knowledge of what is stored in the drive, she could just make an easy search on the internet and take that knowledge. After this is in her hands, the exploitation or not of the driver’s material.

In the offline procedure like above, more keys are required. As we can see in
Table 2, we have three different levels of keys used for the procedure. KTRANS and KMAC are symmetrical keys. Apart from this, the entire procedure of enabling the two entities, on-board system on the train and wayside (RBC), to authenticate, is much longer.

At the beginning, the Key Distribution Center (KDC) [14] creates and distributes a KTRANS to RBCs and trains. This transport key in other words, is used to transfer securely a KMAC at the entities. It is used for both the wayside and on-board entities. Each connection between vehicle and RBC will be performed with a key shared between one vehicle and one RBC (plus the issuing KMC). This is the KMAC, the key that authenticates the entities. It is a symmetric key and its description is explained in the next page. Every time that a train has to communicate with a specific RBC, then another key is created based on this KMAC. This is the KSMAC and is used only for one session. When a KTRANS and a pair of KMAC is created and distributed from a KDC to the train and RBC, then there is no need for more communication with it. Issuing KDC is responsible to transfer the KMAC to the next KDC the train will travel. As Fuloria and others [27] explain, the basic concept behind this is coming from Kerberos [28].

<table>
<thead>
<tr>
<th>Level of Keys</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 Transport Keys (KTRANS)</td>
<td>(KTRANS) is used, in order to encrypt the KMAC key data for the transfer</td>
</tr>
<tr>
<td>2 Authentication Keys (KMAC)</td>
<td>The KMAC keys used to establish a safe connection need to be distributed from a Key Management Centre (KMC) to RBCs and trains (ETCS on-board)</td>
</tr>
<tr>
<td>1 Session Key (KSMAC)</td>
<td>Keys used only for each connection</td>
</tr>
</tbody>
</table>

Table 2: Offline keys hierarchy - Inspired by [29]

An example of the requirements in a KTRANS can be seen in the following line:
"RBC,"+RBC ETCS ID+"_KTRANS_EXCHG,"+GENERATION DATE+".txt"

An actual example of a KTRANS is the following:

Where:
- Variables in italic
- The extension is .txt
- The symbol "+" means concatenation and is not included in the file name
- RBC ETCS DI (values from 00000001 to 16777215)
- GENERATION DATE is YYMMDD
- TNUM is the transaction number (values from 0001 to 9999)
Example:

RBC_00000020_KTRANS_EXCHG_000204_0010.txt

Where:

- **RBC ID**: 20
- **GENERATION DATE**: February 4th 2009
- **TNUM**: 10

KTRANS = <space>02<space>000001<space>000014<space>151208311299<space>02B66E545B62F268C8B6DC9276F7BAE59B7CFDBC857F0D02<space>F8F4898C707CCEAB621FC20437F49D237A70FDEF68830194<space>01<space>AE214D930F18FE68<LF><CR>

Where:

KTRANS = Key Word

02 = KTRANS-EXCHG
000001 = KMC ID 1
000014 = RBC ID 20
151208311299 = Start date is 151208 and end date is 311299
02B66E (...) 02D02 = KTRANS1
F8F489 (...) 830194 = KTRANS2
01 = SNUM 1
AE214D (...) 18FE68 = CBC-MAC

KMC creates a text file that is used for download purposes for the different entities of the on-board and wayside respectively.
4 The Internet

With the full expansion of Level 2 and 3, the insecure procedure of receiving the key by USB drive will come to an end. The new procedure, as is defined by the new standard of online key management, needs only KMACs since the distribution of them will be handled different.

Starting design this project, the question was how to achieve the security between the RBC and the KMC. Different protocols were studied, but the TLS/HTTPS \cite{1}, \cite{30} option was selected, since it is globally used in similar situations and more significantly, it is chosen by the UNISIG consortium of companies, therefore it will be the new standard for every railway around Europe. Some other solutions that could be more secure like Kerberos or SSH \cite{31} were not used even though they could be really good alternatives to TLS. This can be explained, because they were not going to be easy applicable since the TLS is going to be the standard and also because of the special demands from the rail industry. Some examples are the intercommunication among countries and the very fast speed of the trains, something that could create many difficulties in the Kerberos implementation. This protocol demands on clock requirements, which means great synchronization of the entities, something really crucial and not easily achieved in railway industry.

4.1 Description

We all know the word Internet, but what lies behind the word? Internet takes its name from one of its very basic protocols, the Internet Protocol \cite{32}, or also famous with its common abbreviation IP. It is a protocol that exists in the network layer in both TCP/IP \cite{33} and OSI \cite{34} models. An example of these structures can be found in 1. In the figure, two of the most famous models are presented but in real life environment, and in our case the TCP/IP model is used. Especially, since TLS will be used, it is important to describe the exact place of this protocol. It is located, as we can see in Figure 9, between TCP and HTTP protocol, in other words, among Transport and Application layers in TCP/IP model. When the TLS takes place in an application, the layers above it are also secure. An example is the famous HTTPS, which is the common practice for many secure environments.
4.2 TLS

One of the really significant aspects of this project was the implementation of a secure way of communication between the entities. As we described before, TLS protocol was chosen. TLS and its predecessor SSL [36] are two of the most secure cryptographic protocols. Even though SSL is obsolete nowadays, it used to be one of the most secure protocols. It is really common between communications that include servers and clients, such as web applications, email etc. Using TLS, we ensure the confidentiality, integrity, availability and the non-repudiation of the transferred data. Confidentiality, is achieved since, as it is described in the specifications of TLS, the main and primary goals of the protocol are to provide privacy and data integrity between two entities that are communicating above an insecure network. In every message, a message authentication code (MAC) [37] is included and therefore the integrity of the data are ensured.

Moreover, TLS offers authentication of the communication parties, which can only be from the one part of the communication, but as an option could be from both sides and accordingly we will have mutual authentication. At the beginning of every communication the protocol starts with the handshake phase, where the decision that symmetric keys will be used for the session is taken. These keys are based in a shared secret that is exchanged at the handshake face in the beginning.

The authentication between the two entities is done by using Public Key Infrastructure (PKI). An example of this procedure is when someone is visiting a secure website using the HTTPSecure protocol. In Figure 9, the exact place of TLS is given. Apart from this, numerous security issues are known about the protocol [38]. As a reference some of the most known are the Drown attack, the Beast attack, the Poodle [39] and Crime and Breach attacks [38].
Every user has a public key, which is signed by a Certification Authority (CA) [41]. These keys are used to encrypt data, but their main use is to ensure the usage of a shared symmetric key, which will be temporary and different for every session. If a client encrypts her part of the symmetric key with server's public key, only the server can decrypt it and use it, since he has the only valid private key which is connected with the public key. The same procedure is taking place when server wants to send something to the client as well.

When the client is sending something encrypted with server's public key, the only way to be ensured that it is sending to the server that she intend to send is by using digital certificates. The Certification Authority is responsible to create, bind and sign these digital certificates to the legitimate owner. The most common digital certificates are the X.509 [41], an example that was created for this project can be seen in the Appendix 2.
5 Proposed Solution

After careful consideration of the requirements and the procedures, which had to be done, the proposed solution had to handle many different aspects from databases to network protocols and appropriate software for security and networking. All the entities were produced in test environment and specifically were implemented with Ubuntu 14.04.4 LTS [42] virtual machines.

As we can see in Figure 10, the actual components of the project are:

- The Database where the KMACs are stored, in a secure way. MySQL [43], [44] database was configured with three tables inside.

- The Apache Tomcat servlet [45], which contains the actual server written in Java where the RBCs will send their requests and then it will communicate with the database.

- The Apache Web Server [46], which will be react as a proxy server in front of the Tomcat Server and will handle the connections and the requests from the RBCs. This is the server that is handling the TLS/HTTPS connections and it is existence make the whole procedure more secure.

- The DNS server [47]. Since we created our self-signed certificates, we had to create our own DNS to issues certificates. This is a mandatory field of the X.509 digital certificates.

- A Firewall implemented with IPtables [48], [49], [50]. In addition, we used mod-security [51] which will be a Web Application Firewall.

- RBC modules, which are Java programs implemented to liken the procedure of an actual RBC and are sending data to our server in form of testing KMAC.

A simplified view of the project can be seen in the figure below.

![Figure 10: Overview of the implementation](image-url)
5.1 Database

MySQL database was chosen and installed in a Linux Ubuntu server 14.04 edition. MySQL is one of the most used database management systems (DBMS) [52]. It is also used in the LAMP implementation. LAMP stands for Linux, Apache, MySQL and PHP [53]. The creation of three tables it was mandatory, one for the Trains, one for the RBCs and one to store the connection between the Trains and RBCs. The table for RBCs include these columns:

- **id**: which is a unique number to store the items in database and is used only for database administration.
- **R_id**: which is the name of the RBC and it is unique and the primary key of the table.
- **R_final_key**: where the KMACs are stored using a hash algorithm (SHA256) together with a random value (salt).
- **Salt**: the salt number, which is created randomly and is added in the actual key.
- **Date_created**.
- **Date_valid**.

According to this table the table Trains has been created with the following columns:

- **id**: which is a unique number to store the items in database and is used only for database administration.
- **T_id**: which is the name of the Train and it is unique and the primary key of the table.
- **T_final_key**: where the KMACs are stored using a hash algorithm (SHA256) together with a random value (salt).
- **Salt**: the salt number, which is created randomly and is added in the actual key.
- **Date_created**.
- **Date_valid**.

Finally, the table Union, is the one handling the connections between the RBCs and the Trains. So it includes:

- **id**: which is a unique number to store the items in database and is used only for database administration.
- **Rbc_Name**: which is the name of the RBC and is not unique, so we can have it connected with different trains.
- **Train_Name**: the name of the train, which again is not unique and we can see it connected with different RBCs.

None of the tables accept null values for their columns. By having a third table, we achieve the connection between the two tables RBC and Trains. Therefore, when an RBC is searching for a specific train, we can search on this table, if a key pair has already been created between this RBC and this train.

This configuration was chosen, since it protects the database from attacks like brute force [54], dictionary and rainbow tables [54]. The algorithm SHA256 [55] can be changed, and a much slower algorithm (that will prevent from fast cracking) can be used. The way that KMACs are stored, is one of the most common practices in secure environments.

The way that the KTRANS for RBCs and Trains are stored is explained
below. This is a small part of the PHP code created for this reason:

```php
// Inserting of Train id, KTRANS, date_created and date_valid
$escapedName = $POST['fname'];
$escapedPW = $POST['lname'];
$escapeddc = $POST['date_cr'];
$escapeddv = $POST['date_va'];

// Generate a random salt to use for this insertion
$salt = bin2hex(mcrypt_create_iv(32, MCRYPT_DEV_URANDOM));

// Salt is added to the KTRANS
$saltedPW = $escapedPW . $salt;

// Hash method is applied to the salted password
$hashedPW = hash('sha256', $saltedPW);

// Insertion in the database and in the Trains table
$sql = "insert into Trains (T_id, T_final_key, Salt, Date_created, Date_valid) values ('$escapedName', '$hashedPW', '$salt', '$escapeddc', '$escapeddv');

```

There is a PHP [56] implementation, which has all the basic configuration of a database. Some of the features are the login section for the administrator, the add, update and delete options of the KTRANS as well as search of the connections between RBCs and Trains.

PHP is a server-side scripting language and as we described before it is inside the LAMP implementation. It is used in most websites on the Internet on combination with HTML code. In our example the website is only locally accessible to prevent access from unauthorized users.

5.2 Apache server & Apache tomcat

When a client (RBC) sends a request to the server requesting if a train is valid to run in its area, the request first arrives to an Apache Server, who is the responsible to handle the secure communication using HTTPS with the client. After this, Apache server will decide if the client is a trusted entity or not, to decrypt the request and hand over to the Apache Tomcat Servlet.

The procedure is to use an Apache Server as a reverse proxy. The usage of a reverse proxy can vary and there are many situations that it can be useful. Some of the most important are:

- It can be used as another layer of defense, since the servers lying behind him cannot be seen by the outside world.
- Many reverse proxy servers are used as load balancers, which means that distribute the traffic to different backend servers and therefore the servers will never be overloaded and will handle the requests with better speed and high utilization of the resources.
- As it is used in our project, a reverse proxy is used as the server that handles the TLS encryptions instead of the web server. By this, the reverse proxy offloads the web server and let it with its dedicated services.
Another considerable optimization when we use a reverse proxy is that it can serve the static data of a webpage for example. Moreover, it is able to serve dynamic data and as an outcome we have a faster serving time for all the clients [57].

5.3 Servlets

Due to the nature of the project, which is implemented on a test environment, servers Apache and Tomcat are working on the same machine. On real life’s environment there will be more powerful machines that could handle thousands of requests and servers will be on different hardware.

Apache Tomcat is used as a servlet container and enhances the capabilities of the server. In other words, it is using Java servlets which are some of the most common implementations of client-server applications [59]. The advantages of using Servlets comparing of other Common Gateway Interface (CGI) [60] are many:

- One of the strongest advantage is that a Servlet can handle many requests at the same time, in comparison to a CGI.

- Another element that makes a Servlet faster is the way it works. Instead of creating new process for every new request that is coming, it just creates a new thread something that gives it a better performance.

- They have high compatibility. In the most of the implementations the whole servlet can be included in a WAR file and thereafter can easily employed in different applications.

- It can be used in web applications when dynamic data are needed.

- In addition to the previous characteristics, a Servlet is also using the memory efficiently. At the beginning, the Servlet is loaded in the memory and after that it remains there, instead of reloading every time that a request is coming.

- They are also more secure form a CGI, since are server side applications. By this way, are using the security mechanisms of the web servers.
5.4 Certificates and Certification Authority

The reverse server, in our case Apache Server, which handles the connections with RBCs is responsible for handling the secure communications and therefore establishing TLS. TLS as we described before includes some phases, which are mandatory and secure the whole connection. The first that gives the authentication between the two entities contacting is the handshake phase. In our example is like that [62]:

**Client**

Client_Hello

**Server**

Server_Hello

Server_Certificate

ECDH_ServerKeyExchange

Certificate_Request

← ServerHelloDone

Client_Certificate

ECDH_ClientKeyExchange

Certificate_Verify

(ChangeCipherSpec)

Finished

---
We see the ClientHello and ServerHello. Then, the server sends a few other messages, which depend on the cipher suite and some other parameters:

- **ClientHello**: Client starts the connection. As it is specified in the RFC [62] "it is the Client’s job to send a ClientHello at that time". By that time, it means the time after the TCP connection is done between the Client and Server. At that time Client also suggests cipher suites that they can use for the establishment of the connection.

- **ServerHello**: Server cannot send this message prior to Clients hello message. If the ciphers suggested by the Client are not supported form the Server, it will just send an alert message.

- **ServerCertificate**: It is the certificate of the server, which includes its public key.

- **ECDH_ServerKeyExchange** [63]: Additional information send by the Server.

- **CertificateRequest**: In our project we want mutual authentication. After the Server’s chose of the cipher (TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA), Server needs to know that the clients is how intends to be. Therefore, Server is requesting a certificate form the Client as well.

- **ServerHelloDone**: Server sends this message and ensure that finish his sending part. It is an empty message.

The client should verify the Server’s certificate and then reply:

- **ClientCertificate**: Since the Server request for a Client’s certificate, it should send its certificate with its own public key.

- **ECDH_ClientKeyExchange**: Additional information send form the client.

- **CertificateVerify**: This is the verification that the Client is who says that it is, by proving that it has the public key that sent before. It is coming always after the Certificate of the Client.

After the handshake is done, the client and server can exchange data with other higher layer protocols. In this project this protocol will be HTTP. Since TLS connection is established, the HTTP is called HTTPS now, but they have the same structure. One big difference is that of the server side, where most servers handle these connections on port 443 instead of 80 for plane HTTP.

However, the establishment of the TLS connection through the handshake phase includes some certificates exchange. The certifications are needed for authentication of the two entities to each other and it is one of the most common technique for authentication nowadays. All this procedure is also known as
Public Key Infrastructure (PKI). For example, when a user wants to connect to her social account a whole procedure is triggered. Her browser has already stored some of the Certification Authorities (CA) that are legally issued to sign certificates.

These certificates are coming from a trusted CA that is accessibly worldwide. In our project the CA is locally created, since it is a test environment. Therefore, we did not create a verification authority. The CA creates the keys and signs them. Afterwards the keys are transferred to the entities. The need of mutual authentication is mandatory since the server needs to verify that is talking to a valid RBC. It is another layer of authentication of the RBC to the server. Since our project will not be available on the Internet, but it will only be an inside network of the rail industry, the CA will most probably be dedicated for this reason, validation and creation of digital certificates for ERTMS entities.

Therefore, we create the keys in our CA, which is also stored the same hardware with the server and then we manually transfer the certificate of the client (RBC) to Client’s hardware. This is running on a different virtual machine. The whole project will not be available for access in the open Internet. It will be a closed intranet with no access from the outside world.

5.5 DNS

Domain Name System (DNS) is the one of the most basic protocols of the Internet stuck. It has a tree structure and is used to resolve (translate) the IP addresses in normal names, which can be easier usable from humans. It is used from our Client to resolve Server’s IP address and also for the creation of certificates. As it presented in the Appendix 2, certificates require a Canonical Name (CN), which is the authority that create the certificates.

When a certification is created in Java it has to be signed by a CA. This CA could not be just an IP address, so the creation of a local DNS that will issue certificates for the server and the RBCs was the right procedure.

5.6 Firewalls & IDS

The next step is the creation of firewall. This is an optional and basic step. The firewall is a famous practice to secure a server environment. In our system it enables the access at port 443 and port 53 for HTTPS and DNS respectively. At the same time it is installed a Web Application Firewall (mod_security) to secure in a higher level the Apache Server and its application. In addition, the installation of an Intrusion Detection System was examined for the future development of the project. Among others, Snort [64] was chosen and installed since it is open source and really well documented. Right now, the project was created for internal use and this will be an additional requirement for the future. Moreover, project Artillery [65] was installed. Artillery is a python written project and acts as an early detector of attacks in our network and servers. It can enable only specific IPs to establish connection, prevent brute force attacks and even denial of service (DoS) attacks [66]. One of its capacities it enables it to act as honeypot. As a honeypot it opens a variety of open ports in our firewall, which will attract possible adversaries to attack. By this way, another layer of security is added and it is also a way to examine the kind of the adversaries and their attacks.
5.7 Test

A test of the application can be seen in the following figures. By using PSI-PROBE [67], we can have in real time the exact consumption of CPU, memory, the average response time, number of requests and who is sending the requests. It is a really great tool for server observation and statistical analysis.

Figure 13: Number of requests - Created with [67]

Figure 14: Time in (ms) of replying in a request - Created with [67]
6 Results and Future

The final result of the implementation is the secure communication between the RBC (client) and the Server which act as a test Key Management Center (KMC). Since it is a test environment and not the full implementation of the SUBSET-137, the application works with some specific procedure. When the RBC takes a KMAC from a train (on-board), then it sends this KMAC to the KMC requesting if the train is valid to run in its area. As we can see from Figure 14 the requests are computed very fast. In real life environment we expect powerful servers that could handle many requests with the same speed. It is the first time that secure communication is achieved using TLS in railway industry.

Having described the project, we should write some of the steps towards a secure way to transfer and distribute sensitive data in railway industry. First of all, this project should be tested under real data with a lot of traffic. In the near future, all the entities of ERTMS, trains, RBCs and the KMC, will all be connected using TLS. This will create huge load to the servers used in the KMC. Another step should be the testing of the system, of this project and the whole SUBSET-137 in a hostile environment. This will ensure the security of the system and improvements will be done under the experience in this environment. Future development should also be the full creation of KMC/KDC. The operators there should also be able to distribute the keys to every entity of ERTMS requesting a new key. This will demand a great communication between the different companies that consist the UNISIG. Trains, unlike RBCs, are connected with the KMC using wires but they will have to use GSM-R or a more advanced wireless protocol. This will bring security issues, since an adversary could hijack or manipulate the data transferred by the trains, resulting in delays of the schedule or even more serious issues like fatal accidents. The entire project is based in the consideration that it will not be accessed from outside Internet. Further research should be done in order to enable an access from the Internet. In this projects, some further steps toward this direction were investigated. These steps are the installation of Snort, the honeypot artillery and the installation of firewalls.

6.1 Related Work

Previous related work, can be found in resources about railway communications. A very good implementation with the full implementation of Subset-137 can be found from Neat embedded computing [68] company. They implement the project in c-language having open-source code under GNU general public license version 3.0. Another proprietary product is the key.connect [69], which introduces key management system, entities management but the key distribution, the vital point of the system, is still distributed manually. A Slovakian research paper by M. Franekova, P. Chrtiansky [70] examines the birthday attack on the KMAC authentication process. Researchers found that it is possible to have the birthday paradox by using the 3-DES algorithm. Instead, they suggest the future using of AES/Rijndael [71] algorithm. Except of this they describe a detailed path for Key Management System. Following this paper, M.Franekova and M.Vostenak [72] developed a full Key Management System, which explains and covers all the phases, from creation to distribution of the keys. Two older
research papers from Mark Hartong, Rajni Goel and Duminda Wijesekera [73], [74] examine the development of a KDC/KMC in the USA. The basic procedures and standards are the same, but since it is a whole different system than ERTMS/ETCS they applied a similar solution for the USA railway system.

6.2 Sustainability

The implementation of the project and its future development can also benefit environment in a sustainable way. The current procedure, when an update of a key in the system needs to be done, will suspend the traffic in the whole railway system. In economic terms, is not cost-effective since many resources need to aligned in remote areas and the railway industry will suffer from reduces incomes for the time of updating the keys. By enabling the new procedure, cost will be reduced since the update and delete of keys will be easily achieved from a central station, the KMC, with no time delays.
A  APPENDIX 1

In the four screenshots below we can see, one working example captured by Wireshark [70]. We can see how the TLS is established and also the moment that server is communicate with the database.

Figure 15: First part of capture - Created with [75]

Figure 16: Second part of capture - Created with [75]
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.01s</td>
<td>TCP</td>
<td>TCP</td>
<td>1465</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.02s</td>
<td>UDP</td>
<td>UDP</td>
<td>56</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.03s</td>
<td>TCP</td>
<td>TCP</td>
<td>1465</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.04s</td>
<td>UDP</td>
<td>UDP</td>
<td>56</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.05s</td>
<td>TCP</td>
<td>TCP</td>
<td>1465</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.06s</td>
<td>UDP</td>
<td>UDP</td>
<td>56</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.07s</td>
<td>TCP</td>
<td>TCP</td>
<td>1465</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.08s</td>
<td>UDP</td>
<td>UDP</td>
<td>56</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.09s</td>
<td>TCP</td>
<td>TCP</td>
<td>1465</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.10s</td>
<td>UDP</td>
<td>UDP</td>
<td>56</td>
<td>128</td>
<td></td>
</tr>
</tbody>
</table>

Figure 17: Third part of capture - Created with [75]

Figure 18: Fourth part of capture - Created with [75]
APPENDIX 2

This is an example of the keystore created for the project. It can be seen the usage of our DNS (krokos.autun.hom), the certificate chain, the entire certificate, the protocol used (TLSv1.2) and the cipher used (ECDHE-RSA-AES256-GCM-SHA384).

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 2 entries

Alias name: root
Creation date: Jun 24, 2016
Entry type: trustedCertEntry
Owner: EMAILADDRESS=ekrokos@krokos.autun.hom,
CN=ca.krokos.autun.hom, OU=krokos.autun.hom,
O=krokos.autun.hom,
L=Stockholm, ST=Sweden, C=SW
Issuer: EMAILADDRESS=ekrokos@krokos.autun.hom,
CN=ca.krokos.autun.hom, OU=krokos.autun.hom,
O=krokos.autun.hom,
L=Stockholm, ST=Sweden, C=SW
Serial number: 89a56f294518cbb4
Certificate fingerprints:
Signature algorithm name: SHA256withRSA
Version: 1

************************************************************************************

Alias name: tomcat
Creation date: Jun 24, 2016
Entry type: trustedCertEntry
Owner: EMAILADDRESS=ekrokos@krokos.autun.hom,
CN=*.krokos.autun.hom, OU=krokos.autun.hom,
O=krokos.autun.hom,
L=Stockholm, ST=Sweden, C=SW
Issuer: EMAILADDRESS=ekrokos@krokos.autun.hom,
CN=*.krokos.autun.hom, OU=krokos.autun.hom,
O=krokos.autun.hom,
L=Stockholm, ST=Sweden, C=SW
Serial number: a190f9e480b5d0e1
Certificate fingerprints:
Signature algorithm name: SHA256withRSA
Version: 1

************************************************
CONNECTED(00000003)
depth=0 C = SW, ST = Sweden, L = Stockholm, O = krokos.autun.hom,
OU = krokos.autun.hom, CN = *.krokos.autun.hom,
emailAddress = ekrokos@krokos.autun.hom
verify error:num=18:self signed certificate
verify return:1
depth=0 C = SW, ST = Sweden, L = Stockholm, O = krokos.autun.hom,
OU = krokos.autun.hom, CN = *.krokos.autun.hom,
emailAddress = ekrokos@krokos.autun.hom
verify return:1
140289541891744:error:14094410:SSL routines:SSL3_READ_BYTES:sslv3
alert handshake failure:s3
140289541891744:error:140790E5:SSL routines:SSL23_WRITE:ssl
handshake failure:s3
—
Certificate chain
0 s:/C=SW/ST=Sweden/L=Stockholm/O=krokos.autun.hom/
OU=krokos.autun.hom/CN=*.krokos.autun.hom/
emailAddress=ekrokos@krokos.autun.hom
i:/C=SW/ST=Sweden/L=Stockholm/O=krokos.autun.hom/
OU=krokos.autun.hom/CN=*.krokos.autun.hom/
emailAddress=ekrokos@krokos.autun.hom
—
Server certificate
—–BEGIN CERTIFICATE—–
MIID2jCCAsICCQChkPnkgLXQ4fTANBgkqhkiG9w0BAQsFADC
BrjELMakGA1UEBhMCU1cxDzANBgNVBAgTBlN3ZWRlbjEsMB
AGA1UEBxMJU3RvY2ob2xtMHRkwFvYDVQQKExBrcm9rb3Mu3MnY
XVo0d4ua99l9g9tMRkwFwYDVQQLEExBrcm9rb3MnYXVo0d4ua99t
MRswGQQDVQQDFBIqLmtyb2tvcy5hdXR1bi50b20xJzAlBgk
qhkiG9w0BCQEWG1Vc9m9rb3Mna29zLmoundsF1dHVuLmhbT
AeFxw0xNaJ2M1Qx8MzWwMz9aFw0xNa2M1Qx8MzWwMz9aM1GuM
QswCQYDVQQGEwJTVzEPM0GA1UDECBMGU3dIZGtMVRIwFAYD
VQQHEwITdG9ja2lbwG0xGTAxBgNVBAsTEgtyb2tvcy5hdXR
1b5ob20xGTAxBgNVBAsTEGtyb2tvcy5hdXR1b5ob20xGz
AZBgNVBAMUEioua3Jva29zLmF1dHVuLmhvbTEnMCUGCSqGS
lb3QGEJARYYW1tyb2tvc0Brcm9rb3MnYXVo0d4ua99tMlIB
IjANBgkqhkiG9w0BAQEFAOCAQ8AMIIBCgKCAQEA1bw13wP
...
—

SSL handshake has read 1652 bytes and written 138 bytes

New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
Protocol : TLSv1.2
Cipher : ECDHE-RSA-AES256-GCM-SHA384
Session-ID:
Session-ID-ctx:
Master-Key: 297EC20026CB386CD84B2DBAA8D11DF699CBE
7B25A43297A8A7934E9ACDB61721B05EB1337EC0F16A8AB40
570E08E712
Key-Arg : None
PSK identity: None
PSK identity hint: None
SRP username: None
Start Time: 1473355382
Timeout : 300 (sec)
Verify return code: 18 (self signed certificate)
References


[16] *SOLAR-POWERED TECHNOLOGY FOR ETCS INSTALLATION*, Bernhard Stamm, November 2012


[38] RFC 7457 - Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS), https://tools.ietf.org/html/rfc7457


[48] Best Damn Firewall Book Period, Behrens Thorsten, Burlington, 2007


[57] Apache Module Mod_proxy, http://httpd.apache.org/docs/2.0/mod/mod_proxy.html#forwardreverse

