Matti Urhonen

LICENSE MANAGEMENT SYSTEM

LICENSE MANAGEMENT SYSTEM

Matti Urhonen

Master's Thesis

Fall 2014

Information Technology

Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Matti Urhonen
Title of thesis: License Management System
Supervisor: Eero Nousiainen
Term and year of completion: Fall 2014 Number of pages: 67

Anite Finland Ltd. is a Finnish based company providing a full range of software and hardware solutions for the mobile network testing. The Nemo product portfolio consists of several products from handheld measurement tools all the way to the powerful data analysing server solutions. The main customers are mobile operators all around the world.

Anti-piracy and copy protecting is a serious business, as there are skilful people trying to break the copy protections for illegal income or to gain reputation among the hacker scene. This is where the well-functioning licensing tool comes to question.

In this document, licensing is a synonym for copy protection. Regarding the mobile device products, it was decided that the previous licensing method was outdated and was not flexible enough for use of today. There was a serious need for a faster and more efficient way of creating multiple licenses.

When having classified data, it is mandatory to have it encrypted. There are several ways of encrypting the data but some algorithms have gained reputation over others. There are two main categories in data encryption; symmetric and asymmetric encryption. In the symmetric encryption, the same key is used for data encryption and for decryption. The most common symmetric encryption algorithm today is the Advanced Encryption Standard (AES), which utilizes Rijndael-algorithm. In the asymmetric encryption, there are two different keys called the private key and the public key. One key is used for encryption and the other key is used for decryption. One of the most well performing asymmetric encryption algorithms is the RSA.

Cloud services and cloud computing are making their way to the everyday use in the software development business. All major service providers, including Microsoft, Google and Amazon are providing their own solutions for each cloud computing service layers. The “Platform as a Service” (PaaS) -layer is an effective combination of outsourced computing power with a rather flexible software development environment. The Google App Engine is Google’s version of PaaS, and it is available for free and it offers development tools for Python, Java, PHP and Go.

CONTENTS

TERMS AND ABBREVIATION

Term	Description
LTE	Long Term Evolution
NIST	National Institute of Standards and Technology
SHA	Secure Hash Algorithm
AES	Advanced Encryption Standard
DES	Data Encryption Standard
XOR	Exclusive OR
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
TLS	Transfer Layer Security
SSL	Secure Sockets Layer
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
CLR	Common Language Runtime
SDK	Software Development Kit
NDK	Native Development Kit
GAE	Google App Engine
XML	eXtensive Markup Language

1 INTRODUCTION

The license management system was created for Oulu based company Anite Finland (Anite Finland Ltd.) which provides various kinds of solutions for mobile network measurements and testing. The products include measurement software and equipment for smartphones and desktop computers and the product catalog is still growing to meet the need of the increasing number of customers all around the world.

The Nemo product portfolio of Anite Finland provides the customer the full scale of measurement tools from field-testing to data analyzing. Typical customer would be a mobile phone operator launching a new network technology, such as Long Term Evolution (LTE) into use. Another typical use case is when a mobile operator needs to improve the quality of service for the current network and so conduct some measurements in the network area.

This project and this document consider software licensing from somewhat different angle than the software licensing most typically does. In this document the licensing term is mostly used as a synonym for copy protection and anti-piracy tool, as the goal of this project was to create a new, more flexible way of copy-protecting the current and future Nemo products for smartphones. Nemo desktop software licensing methods are not included in this document in any way.

Why licensing and copy protection are so important? As known, companies live to make profit and the money comes from selling products. For some parties software piracy is only a hobby, a way of achieving reputation in the scene of hackers, but for some it is not a hobby but a serious business. Selling unauthorized copies of software for a cheaper price, without a permission of the owner of the software, is an absolute loss of profit for the owner and it can be described as stealing. This kind of illegal behavior can be rejected with proper copy protection. The question is where to draw the line: when does the copy protection come as secure as possible but at the same time will not disturb the

majority of customers, the law-abiding companies and people using the legal copies of the software?

As the first reason for the project, it was known that the previous method of licensing was lacking security and allowing illegal software to be distributed in certain market areas. When being developed at the era of mobile devices without a proper internet connection, it also was quite inflexible in situations where the customer was willing to upgrade the features for their current software and so caused a lot of unnecessary work for the technical support team and also for the customer.

The secondary object was to get to know the current cloud service platforms and choosing the right one for this project. The cloud development done also gave the thesis writer and the team a wide new perspective of what could be done otherwise today, maybe by outsourcing even more of local services to the cloud.

When the project subject was presented to the group of potential end users, the logistic team and the technical support team of Anite Finland, it got a very suspicious and skeptical reception. By the project moving forward and the promises being kept by the development team, the enthusiastic atmosphere of something new and great coming was highly encouraging. That being said, the project was surely one of the favorite moments in the career of the thesis writer at Anite Finland. Hopefully the results will speak for it.

2 DATA SECURITY

In certain situations, it can be mandatory to conceal sensitive information from too curious observers, in particular, when the data cannot be hidden in a more absolute way. The answer to this problem is cryptology, which can be described as science that deals with secret messages that are processed in some manner to make them difficult or impossible for unauthorized persons to read (Salomon, 2003).

First, it is presumably mandatory to explain the need of ciphering and the data encryption, following with the common encryption methods used in this project. This chapter also explains the difference between hashing and encrypting the data. In the context of the document subject, the hash functionality is rather irrelevant as all ciphering is done by encrypting the data, but it is good to recognize the main differences and so get a better grip of the subject.

The chapter also consider the network security, which also plays a big role in the system as all the requests between the different components are done in a secure manner to prevent access of the unauthorized operators.

2.1 Computer security

Ciphering is a challenging operation, which demands fine balancing between security and efficiency. The computer security concept can be divided into three separate sections. The Computer Security Handbook of National Institute of Standards and Technology (NIST) defines these parts as following (Stallings, 1999):

- Confidentiality, which is then divided into two separate parts:
    - Data confidentiality, describing that the private or confidential data is not made available for the unauthorized individuals

    - Privacy, describing that the individuals can control the data related to them. This includes the control of storing of the data and the control of distributing the data only for the authorized individuals.
- Integrity, which is divided into two parts:
    - Data integrity, describing that the information is changed only in a specified and authorized manner.
    - System integrity, describing that the system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation.
- Availability, describes that the system works and is not unavailable for the authorized users.

The NIST standard FIPS 199 (Standards for Security Categorization of Federal Information and Information Systems) lists these three as main security objectives for information systems (Stallings, 1999). This concept is also referred as CIA triad (Figure 1).

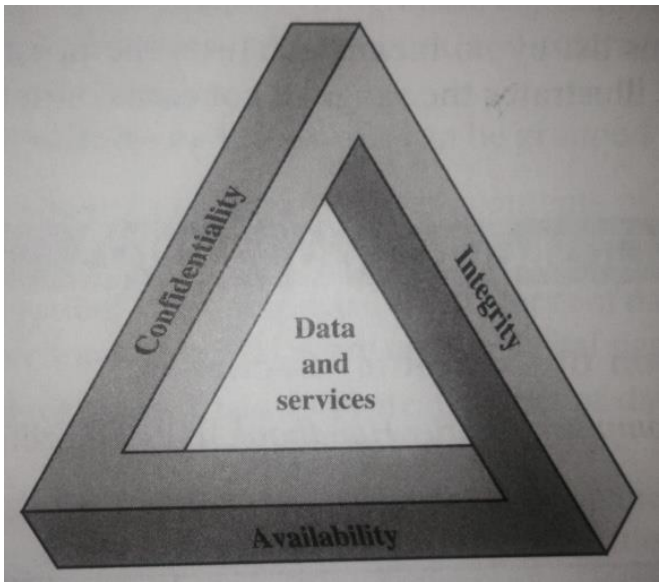

FIGURE 1. The Security Requirements Triad (Pressman, 2010)

As the security objects of the CIA triad are well established, two more objectives have been introduced (Stallings, 1999):

- Authenticity, describing the authenticity of the data. Verifying that the users sending and receiving the data are whom they supposed to be and the data is coming from the authorized source.
- Accountability, the actions on accessing or manipulating the data has to be traceable. Truly secure systems are not yet achievable so the ability of tracing the security breaches is mandatory.

2.2 Cryptography

There are two kinds of encryption methods, symmetric and asymmetric encryption (Stallings, 1999). The symmetric encryption is a simpler approach, having only one secret key compared to the two different keys in the asymmetric encryption, but it is nevertheless no way worse than the asymmetric counterpart.

2.2.1 Symmetric encryption

In the symmetric encryption, the encryption itself and the decryption are done with the same key. The ciphering is done with an algorithm, which uses the shared key in a certain manner to achieve the ciphertext as an output from the input plaintext. (Stallings, 1999.)

The symmetric encryption, or the single-key encryption was the only kind of encryption prior the public key encryption and is still the most widely used encryption method (Stallings, 1999). Symmetric encryption consists of five ingredients:

- **Plaintext:** The main input which is to be encrypted
- **Encryption algorithm:** The algorithm which performs various operations to cipher the plaintext
- **Secret key:** The second input for the algorithm. The algorithm will produce a different output form the same input depending on the key, the ciphertext and the decryption algorithm.

- **Ciphertext:** The output of the encryption algorithm, a scrambled message. The ciphertext depends on the plaintext and the key and the ciphertext is unique for each key provided. At the same time, with the same input and the same key, the ciphertext is also identical.
- **Decryption algorithm:** Essentially the encryption algorithm run backwards. It takes the ciphertext as an input and uses the key and the algorithm to provide the plaintext as output.

These five ingredients can be seen as steps in Figure 2.

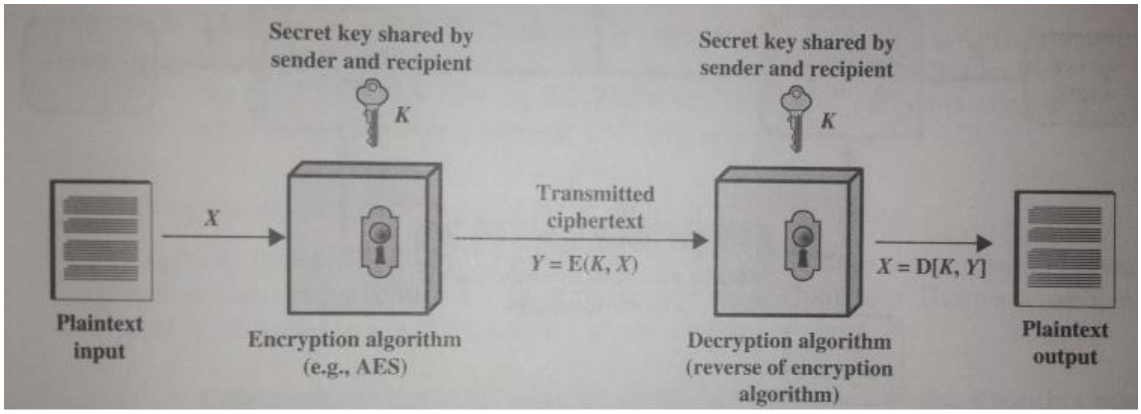

FIGURE 2. The simplified model of the symmetric encryption (Stallings, 1999)

2.2.2 Asymmetric encryption

The asymmetric encryption relies on the security coming from two separate keys. The encryption and the decryption are both done with different keys, one with the public key and the other with the private key. (Stallings, 1999.)

The keys are created as pairs, when one changes the other has to be changed. This is possible because the public key, as the name defines, is allowed to be public and there will be no security concerns from this. The private key though is demanded to be protected and secret from everyone else but the owner. (Stallings, 1999.)

The need for the asymmetric encryption raised from two problems concerning the symmetric encryption scheme. The fact that when having encrypted communication between recipients, the key had to have already been shared in some way or there was a need for some kind of key distribution center. Pre-sharing the key can be clumsy in certain situations and especially when the key has to be changed. The pioneers of the asymmetric cryptography, Martin Hellman and Whitfield Diffie also despised the key distribution center approach, as that would ruin the very essence of secret messaging; keeping the communication totally secret from the unauthorized individuals. (Stallings, 1999.)

Other main reason for the development of asymmetric encryption was a need of reliable digital signature, which could be compared to the traditional handwritten signature on paper documents (Stallings, 1999).

Asymmetric algorithms share a common characteristic, where the decryption key is computationally infeasible to determine by knowing only the cryptographic algorithm and the encryption key (Stallings, 1999). The asymmetric encryption process is visualized in Figure 3. Also in Figure 3, there are introduced all the six components of the encryption process:

- **Plaintext:** The readable input message to be encrypted
- **Encryption algorithm:** The algorithm which processes transformations to the plaintext
- **Public key and private key:** Two separate, unequal keys, which are created as a pair and are used as a pair where one encrypts and the other decrypts the message.
- **Ciphertext:** The message encrypted with the public-key of the receiver. Ciphertext can be decrypted only with the right algorithm and the correct private-key
- **Decryption algorithm:** The algorithm, which decrypts the ciphertext using the private-key of the receiver.

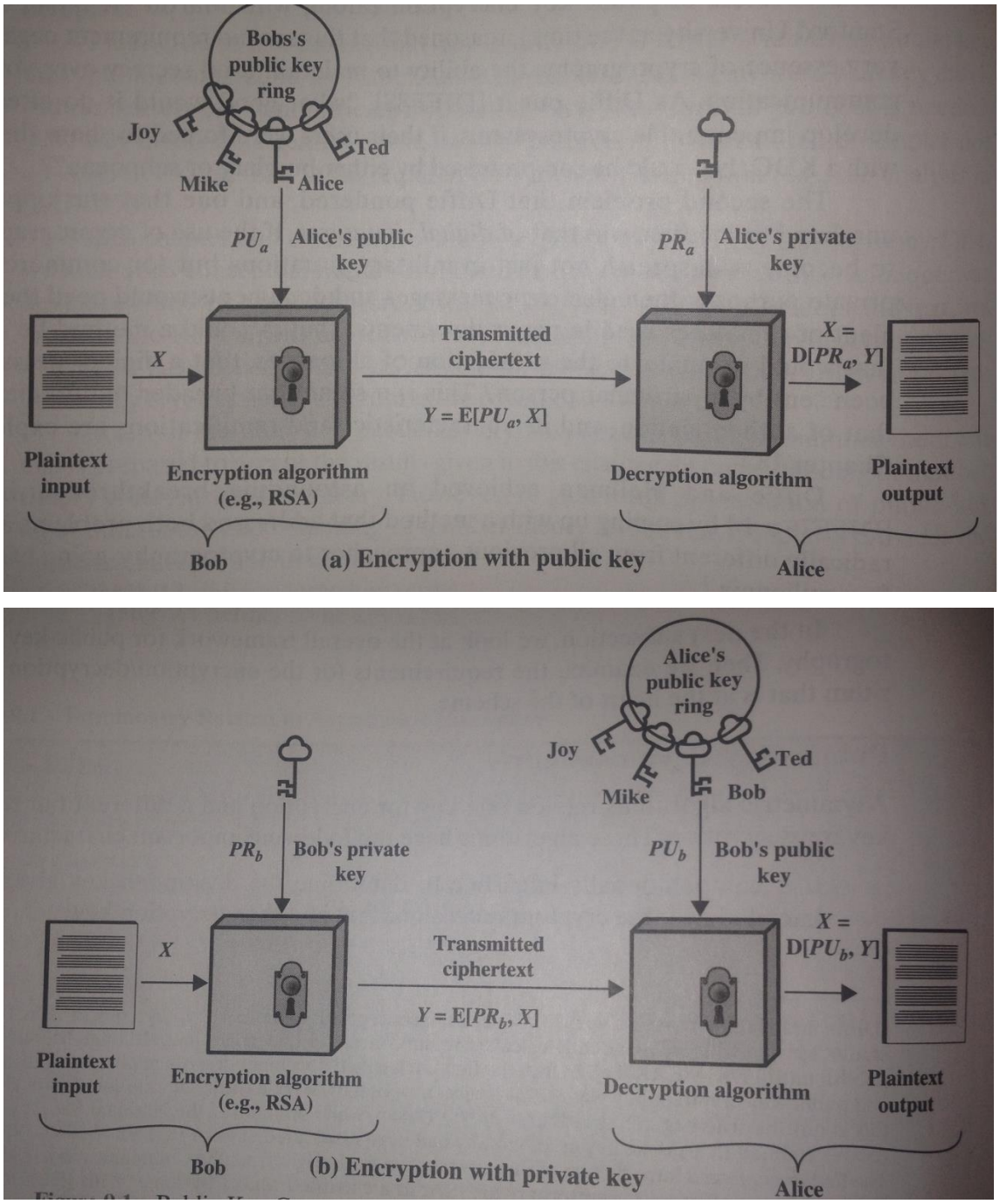

FIGURE 3. Public-key cryptography (Stallings, 1999)

Differing from the messaging use of the asymmetric encryption, in the digital signature use the document is encrypted or signed with the private key of the sender. Signing is usually done to a small block data and then the signature can then be verified by using the public-key of same key pair to decrypt the data block.

2.2.3 Hashing

When the data encryption enabled the ciphered text to be decrypted to plaintext again, hash functions operate differently. A hash function accepts a variable length data block as an input and produces a fixed-size hash value as output. A hash from same input is always same, but unlike with the encrypting, the hash function is irreversible. This kind of cryptographic feature is used in data comparison for seeing if the data has changed or not. (Stallings, 1999.)

One of the most famous hash algorithms is the Secure Hash Algorithm (SHA), which was developed and published by NIST as a federal information processing standard in 1993. While many other hash functions have been breached during the years, SHA has kept its place as a secure hash function with help of version updates. (Stallings, 1999.)

There are free tools for creating and testing different hashes. In Figure 4, there is an example of a hash created from a clear plaintext input with online SHA-1 converter (SHA-1 online, 2014).

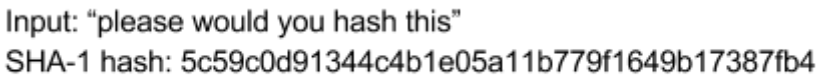

FIGURE 4. Example of SHA-1 hash (SHA-1 online, 2014)

Now, when utilized to use for example as a password, the server would not contain the plaintext version of the text at all, but only the SHA-1 hash version, which it would then compare to the hashed user input.

2.3 AES

In 1997 US National Institute of Standards and Technology (NIST) announced that a new standard for data encryption will be developed, named Advanced

Encryption Standard (AES). It would then replace the old encryption standard called Data Encryption Standard (DES) and triple-DES (Daemen, 2002).

The selection process for the new encryption standard differed from the previous standardization processes so that the process would be open and anyone could provide a ciphering candidate and so each submission, which met the initial requirements would be evaluated by NIST (Daemen, 2002). Finally, a winning algorithm candidate, Rijndael, was selected and the new standard AES was published in 2001 (Stallings, 1999). The name of the algorithm comes from a combination of the algorithm designer surnames, Vincent Rijmen and Joan Daemen (TechTarget, 2014).

2.3.1 Operation

AES, as all other encryption methods, involve arithmetic operations on integers such as addition, multiplication and division, on 8-bit bytes. AES allows keys with sizes 128, 192 or 256 bits and the key length also defines the name of the used algorithm, AES-128, AES-192 or AES-256. (Stallings, 1999.).

The general structure of the encryption process is visualized in Figure 5. The algorithm takes the input as 16-byte (128-bit) blocks, which are divided into 4 x 4 matrix of bytes. Also, the key is transformed into matrix form (Stallings, 1999).

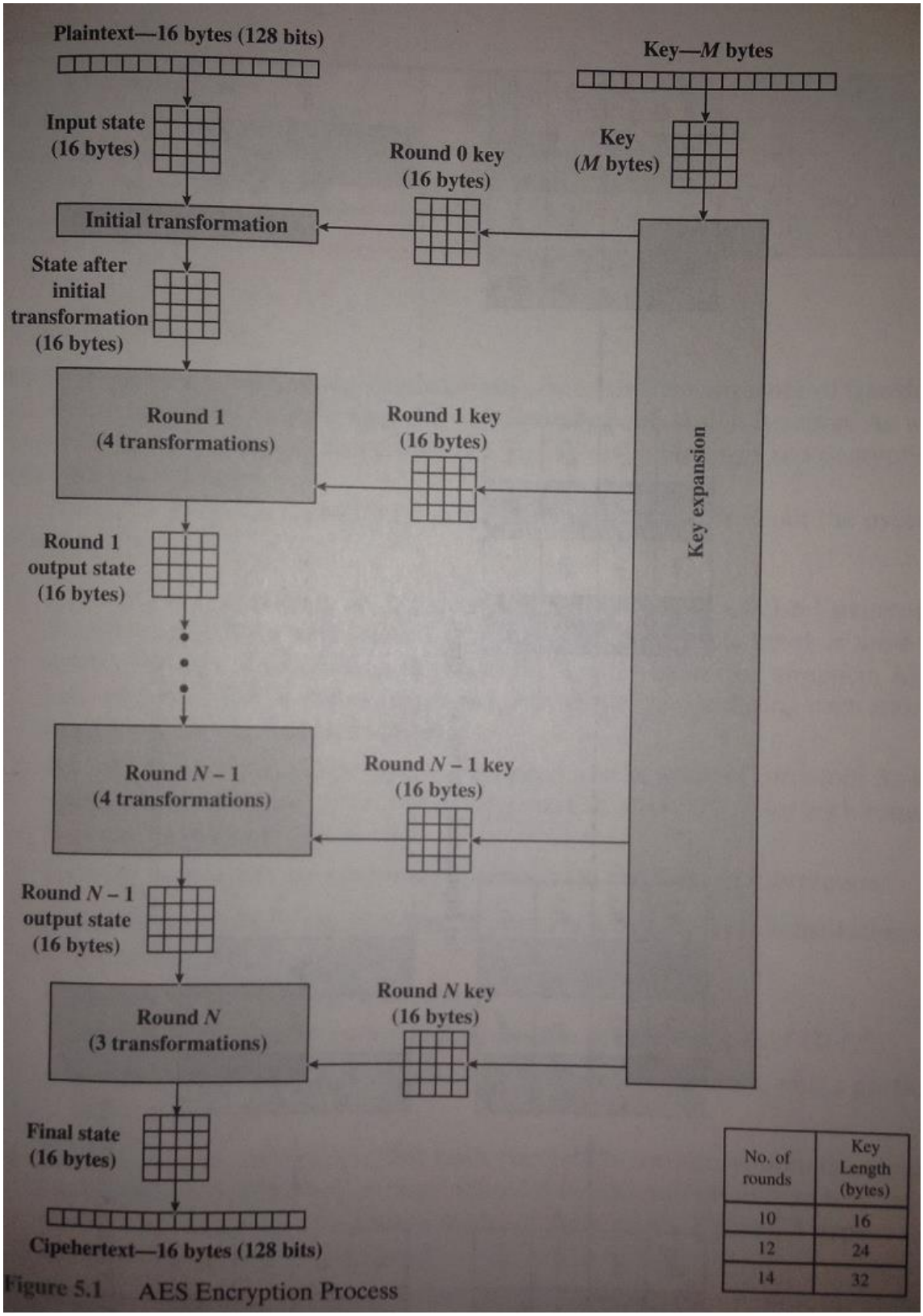

FIGURE 5. AES encryption process (Pressman, 2010)

The input blocks are processed through initial transformation and then copied to the state array of same size, which eventually is the object of each stage or round of the encryption or the decryption process. The initial transformation consists of only one operation, AddRoundKey. The first N - 1 rounds consist of four different transformation functions of the AES, called SubBytes, ShiftRows, MixColumns and AddRoundKey. The total number of the rounds is defined by the key size, with 128-bit key there are 10 rounds, with 192-bit key there are 12 rounds and with 256-bit key there are 14 rounds. The final round consists of only three transformations. (Stallings, 1999.)

The transformation functions are the core of the process as they do the modification of the input. Of the four transformation functions, only the AddRoundKey function uses the key. The other functions operate different arithmetic operations such as Exclusive OR (XOR) and byte-to-byte substitutions. (Stallings, 1999.)

2.4 RSA

Developed by Ron Rivest, Adi Shamir and Len Adelman (R.S.A.) in 1977, the RSA was one of the first successful asymmetric encryption methods and still keeps a reputation of the most widely accepted implementation of the public-key encryption (Stallings, 1999). RSA is widely used as a tool for securing the data sent over the Internet. It is the base for many popular security protocols such as the SSH and the SSL/TLS (TechTarget, 2014).

In RSA, both public and private key can be used for encryption when the opposite key does the decryption. RSA is providing its users the key points of computer security, confidentiality, integrity and authenticity thus maintaining the popularity among the data transfer security scene. (TechTarget, 2014.)

The security of the RSA algorithm is based on factoring large integers that are products of two large prime numbers. As computing power of supercomputers increase, the security of the calculation-based algorithms also weakens. This is

due the fact that the security comes from the aspect of how long the cracking time would be comparing to the benefit of the results. By reducing the time needed for the cracking calculations, the benefit rises. (Stallings, 1999.)

An example of using the RSA algorithm is presented in Figure 6. This example has helped the thesis writer to understand the mathematical aspect of the RSA algorithm and the public-key encryption in general.

Alice generates her RSA keys by selecting two primes:

$p = 11$ and $q = 13$.

The modulus:

$n = p \times q = 143$.

The totient of n:

$\phi(n) = (p-1) \times (q-1) = 120$.

She chooses 7 for her RSA public key *e* and calculates her RSA private key using the Extended Euclidean Algorithm which gives her 103.

Bob wants to send Alice an encrypted message *M* so he obtains her RSA public key (*n*,e) which in this example is (143, 7).

His plaintext message is just the number 9 and is encrypted into ciphertext *C* as follows:

$M^e \bmod n = 9^7 \bmod 143 = 48 = C$

When Alice receives Bob's message she decrypts it by using her RSA private key (*d*, *n*) as follows:

$C^d \bmod n = 48^{103} \bmod 143 = 9 = M$

FIGURE 6. Basic RSA example (TechTarget, 2014)

2.5 Internet security

To understand the main aspect of Internet security of this project, the Hypertext Transfer Protocol Secure (HTTPS), a little knowledge about the Hypertext Transfer Protocol (HTTP) is required.

2.5.1 HTTP

The HTTP is the base of data transfer in the World Wide Web as each request is transferred by the rules defined by the protocol. The HTTP gives a standardized way for the computers to communicate with each other over the network (Wong, 2000).

When considering the use of the HTTP, it can be surprisingly familiar from the everyday Internet browser use. As the user writes the address, it already contains information of how to construct the HTTP request. For example, the address http://hypothetical.ora.com:80/index.html contains several different parts described below (Wong, 2000):

- **http:// :** The protocol to be used for the request. This can be https:// or some other protocol known for the applications used.
- **hypothetical.ora.com:** The address or the hostname of the server to be connected and requested.
- **:80 :** The port number to be used. The port 80 is the default port of HTTP transfers and usually it is not needed for the browsers to work correctly.
- **/ :** The path after the hostname and optional port number defines the resources requested from the server.

From that information, the web browser constructs a HTTP request, which is illustrated in Figure 7.

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/
    jpeg, image/pjpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE
    5.01; Windows NT)
Host: hypothetical.ora.com
Connection: Keep-Alive
```

FIGURE 7. HTTP request from browser to server (Wong, 2000)

The server receives and processes the request. The server tries to find the resources requested by the client from the location specified in the address, in this case the index.html. When it is ready and all the data is collected, the server sends a response to the client, in this case to the web browser. The response is illustrated in Figure 8.

```
HTTP/1.1 200 OK
Date: Mon, 06 Dec 1999 20:54:26 GMT
Server: Apache/1.3.6 (Unix)
Last-Modified: Fri, 04 Oct 1996 14:06:11 GMT
ETag: "2f5cd-964-381e1bd6"
Accept-Ranges: bytes
Content-length: 327
Connection: close
Content-type: text/html

<title>Sample Homepage</title>
<img src="/images/oreilly_mast.gif">
<h1>Welcome</h1>

Hi there, this is a simple web page.  Granted
it may not be as elegant as some other web
pages you've seen on the net, but there are
some common qualities:

<ul>
  <li> An image,
  <li> Text,
  <li> and a <a href="/example2.html">
hyperlink. </a>
</ul>
```

FIGURE 8. HTTP response from the server (Wong, 2000)

With the request as the one used in this example, http://hypothetical.ora.com:80/, the server returns the resources associated with the root path and returns the data. Usually this is a synonym with the index.html file or some equivalent. (Wong, 2000.)

The response is divided into two definitely separate parts, the header part and the content part. The most important part of the header considering this document is the response status code, in this case the 200 OK, which informs the client that the requested resources have been found and are transmitted in the response. This being said, by the HTTP response status code the client can define if the

request was successful or not (Wong, 2000). The most usual and most well-known status codes for failed requests would be the “400 Bad request” and the “404 Not Found”.

The content part is the HTML data sent from the server. The HTML code in Figure 8 would be rendered as a valid web page with a picture, a text paragraph and some list items.

2.5.2 HTTPS

The Internet has a ton of security risks for a neglectful user. The requested server can be infected with malware or Trojans, which are then sent along with the responses to the client computer, or the network traffic can be spied if not encrypted and so valuable information can be caught by thieves. The answer to some security threats can be encrypted communication between the client application and the web server (Stallings, 1999).

The Hypertext Transfer Protocol Secure (HTTPS) is a combination of HTTP and the Transfer Layer Security (TLS). HTTPS allows the client application, for example a web browser and a web server to have share a secure, encrypted line for communication (Stallings, 1999). By the viewpoint of a web browser user, the main difference compared normal HTTP communication is that the protocol at the beginning of the address is typed “https://” instead of the “http://”. Also, the HTTPS uses the port 443 as default (Stallings, 1999).

SSL/TLS

The security of the HTTPS is based on the Internet security standard called TLS. TLS is built over the Secure Sockets Layer (SSL) cryptographic protocol, which bases the security on multiple different aspects, including the RSA and Diffie-Hellman public-key algorithms. The first version of TLS was branched from the SSL version 3, as there was a need for a specifically standardized security protocol for Internet. The first version of the TLS can be viewed as SSL version 3.1 as it is very close and backwards compatible with SSL version 3 (Stallings,

1999). The name Transport Layer Security comes from the fact that the security is built on top of the transport layer TCP (Figure 9).

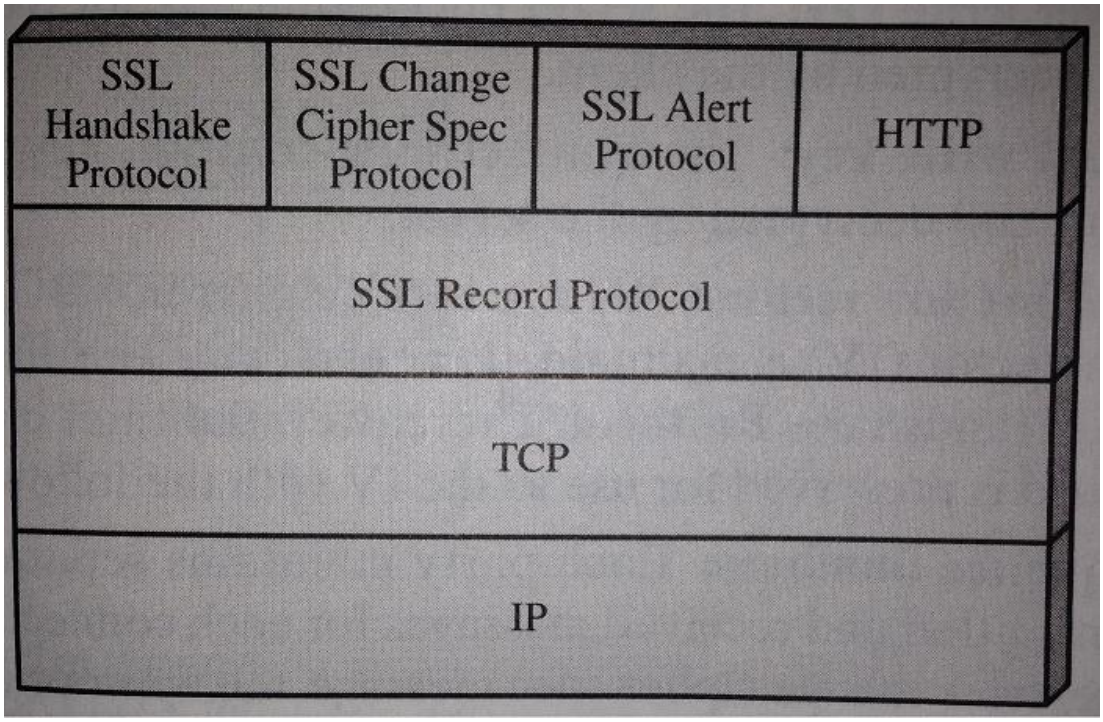

FIGURE 9. SSL protocol stack (Stallings, 1999)

HTTPS communication

The client begins the secure connection by connecting the server on the appropriate port and sends the TLS ClientHello packet to begin the TLS handshake. As the handshake is completed, the client can send the first HTTP request. In the HTTPS communication, following elements are encrypted (Stallings, 1999):

- URL of the requested document
- Contents of the document
- Contents of browser forms
- Cookies sent from the browser to server and from server to browser
- Contents of HTTP header

The connection is being closed after the client sends the regular HTTP connection closing command. Also the underlying TLS session requires closing at the server side. Usually the server shuts the TLS connection down in a controlled manner and sends the notification of this to the client, but it can be done in such a harsh way that the client will receive an “incomplete close” notification. As this could be a sign of a programming error or of a possible attack against the connection, HTTP clients are expected to cope with this kind of situation where the server has done something unexpected to maintain the security. (Stallings, 1999.)

3 SOFTWARE ARCHITECTURE

When having no experience of software development or the code writing process, it can be rather strange to use the words architecture and software in the same context. While the other word is usually strongly united with buildings of various kinds and the other usually means an interaction happening on the screen, they still have a very real connection between them.

Speaking of buildings, software architecture can be compared to the architecture of a building (Pressman, 2010). When thinking of architecture as in its simplest form, a house can be imagined with a certain kind of façade, with a specific shaped roof and finally with some specific kind of wall material. Architecture, even with just buildings, is nevertheless a lot more. It consists of the instructions of how every piece of, in this case, the building are connected and related to each other forming a well functional ensemble. This kind of mentality can be transferred into the world of software architecture.

The software architecture itself is not an operational software but it is or it gives a clear guideline of where different software modules should locate, what is their role and how and with whom they should communicate. In their book “Software Architecture in Practice, Addison-Wesley 2003”, Bass, Clement, and Kazman say that “the software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them”. (Pressman, 2010.)

What this means is that the software architecture includes all the aspects of the functional software. From the visible User Interface being the facade and the material of the walls, to the modules under the visible layer being the household mechanics as the air conditioning and the heating system, and last but not the least, all the communication between the modules and the User Interface being the electricity and the water network inside the structure.

As leaving the building metaphors aside and approaching software architecture purely from the software developers side, it can be described as a process of defining structured solution that meets all of the requirements, while optimizing the system for performance, security and manageability (Microsoft, 2009).

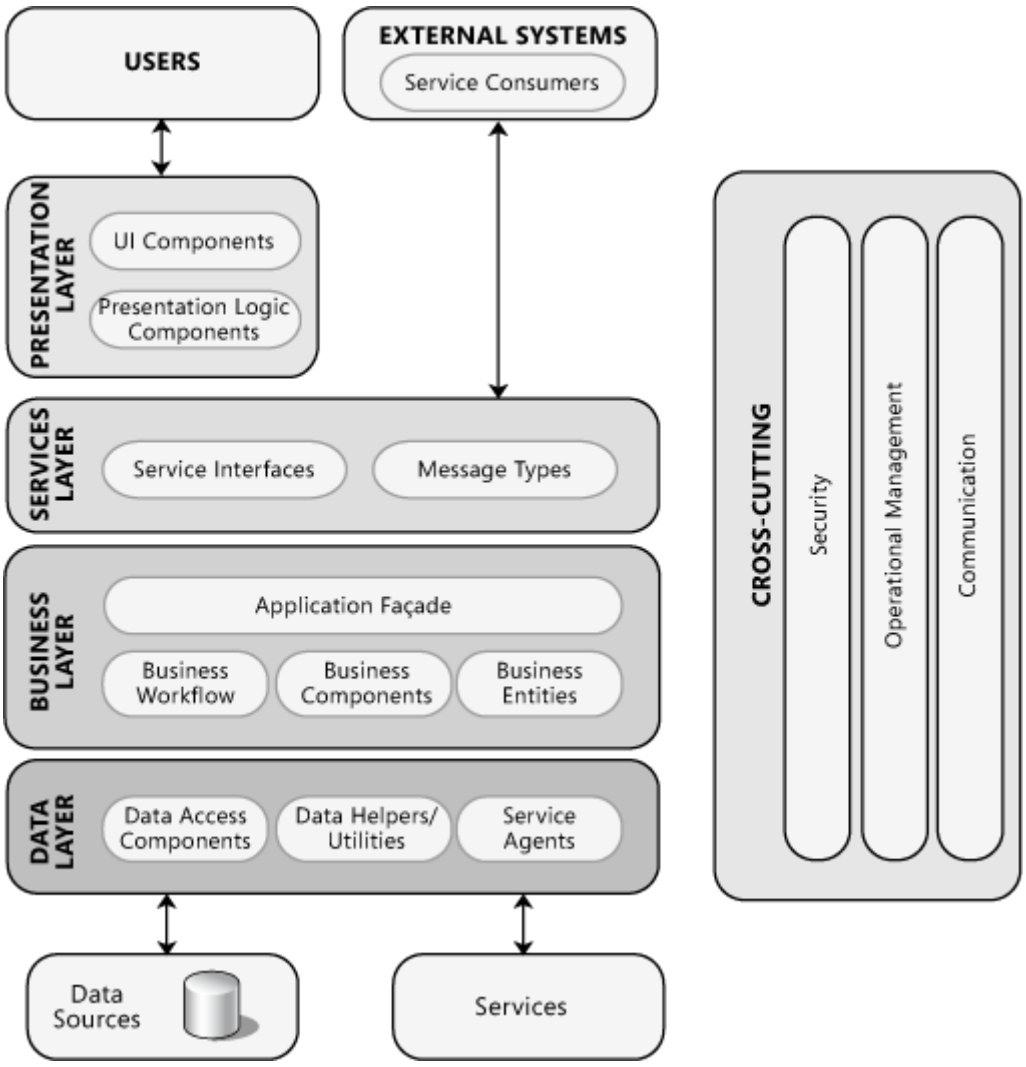

FIGURE 10. Software architecture representation (Microsoft, 2009)

Importance of the software architecture can be identified with few key reasons. The representation of the software architecture, for example as seen in Figure 10, can be the only way of enabling all the project participants to communicate with each other. While the customer has no information of software development, they still can get a grasp of the final product from a proper architecture design. At the same time it can be an effective way of finding the most efficient ways of

developing the software instead of the all too common style of developers rushing into action. (Pressman, 2010.)

3.1 Client-Server architecture model

The architecture model, style or pattern is a set of principles that provides a framework or a shape of an application. Architectural approach improve software modularization and promote the design reuse by providing solutions to the most common development problems, which are faced during various projects. (Microsoft, 2009.)

Understanding of architectural styles provides several benefits. It gives a common language and helps to allow a conversation between, for example, the software developer focusing on specific solutions and the client focusing on more trivial aspects of what the software should be able to do. (Microsoft, 2009.)

In the client-server architecture, the client represents the user in need for some service provided by the server. A client makes a request, the server receives and processes the request and sends back the required response (Oracle, 2001). The simplest model of client-server system includes a server application that is accessed directly by multiple clients (Microsoft, 2009).

The client-server architecture has a pedigree of being the one of the most common software architecture in business or school computer systems. In these systems the client represented a user interface application running on a desktop computer while the intelligence and the data was located on the server side and was accessed by multiple clients at the same time (Microsoft, 2009). Not much has changed since those “central computer” days considering the architectural side of today as one of the most common client-server software types would be a web-browser based application accessing the server data with the browser being the client user interface.

In Figure 11, the clients are accessing the server through internet, but the connection could be arranged so that the clients access the Internet through the server, being in the same local area network with se server.

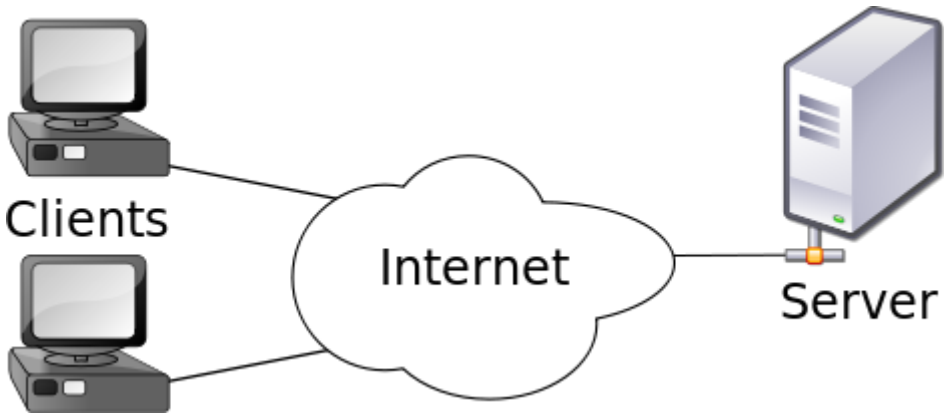

FIGURE 11. Client-server architecture

The basic characteristics of the client/server architecture model include (Oracle, 2001):

- **Asymmetrical actions:** The server receives requests from multiple clients seemingly simultaneously and processes the requests parallelly. The dialog is always initiated by the client and the server passively waits for incoming requests.
- **Encapsulation of services**: The server is the specialist which has to be able to fulfill all the requests from the clients. Servers can be upgraded without affecting the clients as long as the messaging layer is kept unchanged.
- **Data integrity:** The data is centrally maintained which can result in cheaper maintenance and improved security aspects.
- **Platform independence:** The ideal client-server situation is independent of hardware and operating system restrictions, allowing the mixed platform communication.
- **Separation of functionality:** Even when run at the same physical computer, the client and the server both have their own functionalities, which should not overlap.

4 CLOUD SERVICES

Today modern companies are seeking for efficient and low cost possibilities for offering effective services for customers. Maintaining the hardware infrastructure can be difficult when the demands of the service may differ a lot during the development phase and especially during the time the service is active. The more the clients use the service or the application evolves more complex and demanding, the more computing power is needed. Also the size of the data storage needed can increase rapidly in some situations. When using cloud services all these problems can be outsourced to the cloud service operator. This comes also with the benefit that the company only has to pay for the resources that are being used and the cost goes up only when more resources are needed.

Cloud service providers offer solutions in different levels from infrastructure layer to the application layer. Customers can have only the necessary computing power with operating system images, on which they can build their own platforms and applications for their needs. On the other side of the service model spectrum, there are predesigned online applications ready to be tailored for the need of the end customers. (Salo, 2012.)

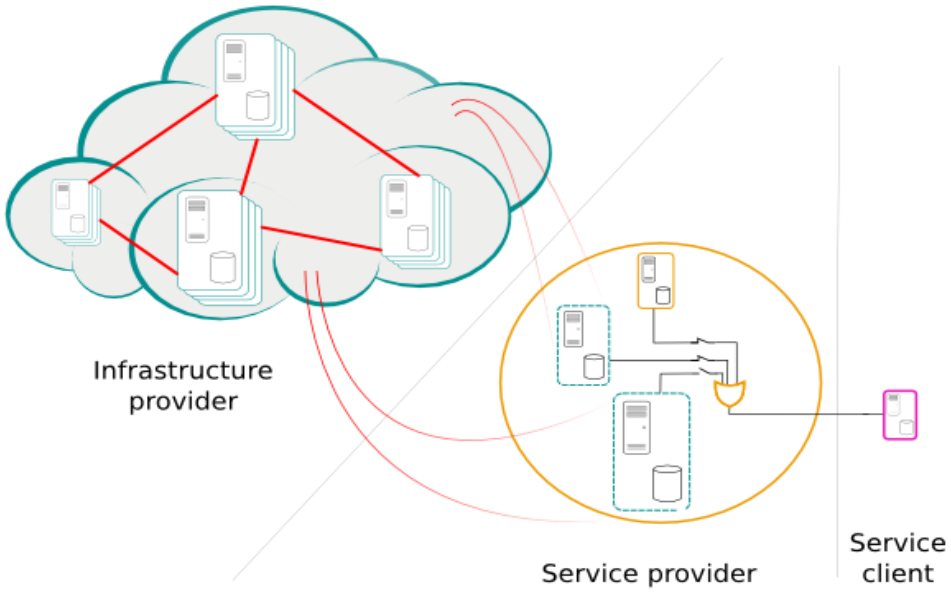

FIGURE 12. Cloud demonstration

4.1 Cloud services

For an average Internet user the term ‘cloud’ might be a quite abstract word, not describing the full functionalities which are provided nowadays by major cloud service providers as Google, Microsoft and Amazon. The most well-known example of simple cloud service would be an advanced file storage where the user can upload files or even synchronize a part of the local hard drive of the personal computer. The owner of the original file can then give the other cloud users privileges to view and modify the files and the modifications can then be synchronized to the local files. For example, Google Drive and Dropbox are well known for this kind of file sharing feature.

When exploring the cloud services a bit deeper, it can be found that the cloud service can mean a lot more. Some of the cloud service providers offer such functionalities as text editing, spreadsheet and slide show tools which consumers are used to use directly as standalone applications in a local computer. This kind of service model, the “Software as a Service” (SaaS) cloud service model, which offers online application operating inside of users web browser, brings the average Internet user closer to the true possibilities of the cloud services. (Salo, 2012.)

4.1.1 Advantages

The advantages for companies of bringing the cloud services in to use come from the economic aspects. When having no need for heavy construction for hardware and network infrastructure at the beginning lowers the start costs of the company with a benefit of the fact that the cloud solution will also be a very strong and lasting choice if done correctly. (Salo, 2012.)

Considering the technological advantages of the cloud services, the main aspects that differs the cloud from being a hosted server are the scalability and the cloud applications.

Scalability

In a situation where a small company is eventually launching a new product, the demand of the hardware and the network resources can be quite low. In this case, it can be attractive to build as cheap system as possible and worry about the insufficient resources later. This, of course, is the wrong approach and leads to potential loss of current and new customers.

The scalability feature of the cloud services ensures having the sufficient system from the very beginning and not having to pay for something that is not being used. During the life cycle of the service the demand of resources can expand rapidly and in this situation the maintenance of an network infrastructure might be even impossible. When outsourcing the architecture to a cloud service it is most presumably ensured that the computing power and the hard drive space needed will meet the requirements. (Salo, 2012.)

Applications

The relatively easy learning curve and the variety of service models give a different approach for the software development companies and especially for the non-software companies to bring customers closer with smoothly running mobile applications. The secret of the so-called smoothness lies in the outsourced computing power while giving the device itself only the necessary tasks to bring information on the screen. When the creation and the upkeep of the software and the database does not anymore need dedicated servers and deep understanding of software development, it is easy to recognize the potential of the cloud applications. (Salo, 2012.)

4.1.2 Disadvantages

As in every system, there are some risks and disadvantages when using cloud services. The most notable risk is security. When the valuable data of a company is located somewhere outside of the local area network and firewall, it requires some consideration of what to store in the cloud. The recent larger information

network intrusions reveal that many service providers do lack in network security, so absolute security is not given. (Salo, 2012.)

Another disadvantage is the dependency of Internet connection. When providing services in a hosted network environment it is essential that the connection between the provider and the customer is working properly. Internet traffic is vulnerable for various kinds of attacks and it is possible that for some reason the connection may be restricted to specific services. (Salo, 2012.)

The pricing is generally speaking an advantage, but it can cause a risk. Although the start cost of using cloud service can be attractively low, the cost might add up being more than it could be when having an own static infrastructure (Salo, 2012). These variations are much related to a specific situation because the pricing of cloud services vary much and different kind of operations from data storage size to downlink traffic are priced separately (Google, 2014).

4.2 Cloud computing

When exploring the cloud services more deeply it is apparent that it means more than a basic file hosting service. The cloud computing happens in a system where a large number of computers are connected as a network which enables them to communicate with each other and simultaneously process tasks assigned to them as a group. By creating this kind of computing clusters, the computing power is very capable. By the feature of scalability, it is possible to increase the power needed for a specific application and so meet the requirements even when the same application in the same cloud service evolves. (Salo, 2012.)

The cloud computing services can be divided into separate layers or service models. The service models can be represented in a simple hierarchy model (Figure 13), which describes the physical location of the model.

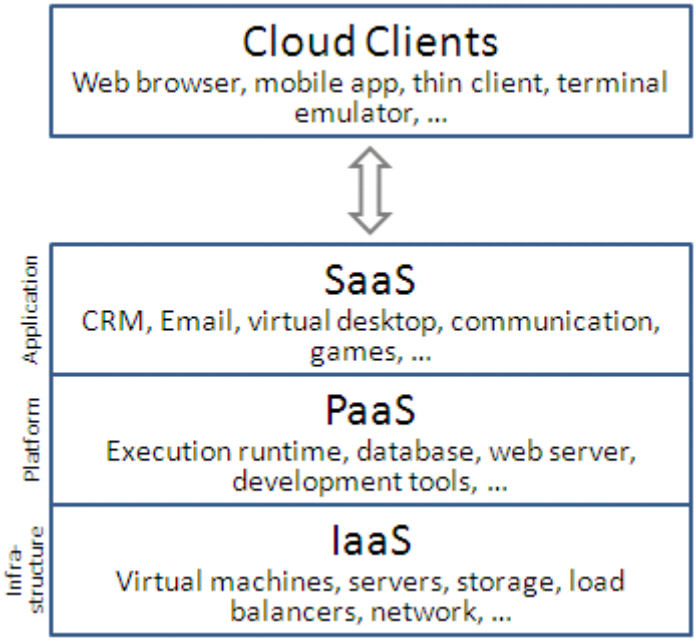

FIGURE 13. Cloud service models

4.2.1 Cloud client layer

On the top of the service model stack is the client layer. This usually consists of a web browser running on an Internet enabled device while it can also be a customized application, which can interact with the cloud service. The most common cloud client devices are a personal computer or a modern mobile device with a proper Internet browser. In mobile device use, the cloud services are mostly accessed with customized applications, which tend to provide a more efficient user interface compared to the web browser view. (Salo, 2012.)

4.2.2 Application layer

The second layer is the application layer consisting of the "Software as a Service" (SaaS) service model. The company utilizing this service model relies on the software, platform and the infrastructure provided by the cloud service provider, which reduces the development costs but at the same time restricts the possibilities of use cases. Most notable examples of this layer are the commercial cloud services such as Google Drive or Google Docs, which provides the user the file storage with common office applications like a text editor and a spreadsheet tool. (Salo, 2012.)

4.2.3 Platform layer

The third layer is the platform layer consisting of the “Platform as a Service” (PaaS) service model. It differs from the application layer so that the user is allowed to develop and run their own applications in the environment managed by the cloud service providers. The cloud service provider gives the development tools and the means to upload the application to the server. (Salo, 2012.)

4.2.4 Infrastructure layer

The fourth layer is the infrastructure layer consisting of the “Infrastructure as a Service” (IaaS) service model. In this model the user is responsible of the full operating system including the platform and the applications running on top of it. The cloud service provider only provides the infrastructure where the virtual machines are executed. The cloud service provider can provide operating system images from which the user can select the most favorable. (Salo, 2012.)

4.3 Platform as a Service

While outsourcing the hardware infrastructure to the cloud service provider, it is possible to implement cloud applications running on top of the scalable cloud architecture with almost unlimited data storage. This service model is called the “Platform as a Service” service model. It enables the application developers to quickly implement and deploy applications to the cloud with no dependencies of different operating systems or platforms.

In commercial aspect, the “Platform as a Service” service model basically means that the customer rents the computing power and the prebuilt operating system environment from the virtual machines running on top of the computer cluster located in a server center of the service provider. The rented environment consists of a cloud management interface, development libraries and an interface, which is used to upload the applications to the cloud system. (Salo, 2012.)

4.3.1 Development tools

The service provider is responsible of enabling the user to get the needed solution development kits and other development tools such as development plugins for common integrated development environments (IDE). Usually the solution development kit comes in multiple different programming languages, which eases the learning curve. The variety of supported platforms can be a necessity while the competition from the customers between cloud service providers is ongoing with accelerating speed.

The development tools can include means for such operations as the data store access and queries, the HTTP(S) request handling and sending email. While these operations are implemented in the application programming interface of the cloud service, there can still be many more libraries providing the more advanced language specific functionalities. (Salo, 2012.)

4.3.2 Advantages

Considering the advantages of switching from normal locally ran applications to hosted Platform as a Service solutions, the most efficient aspect would be that there is no need to keep the client applications up to date and no fear of having miscommunication between the client and the server. The software itself is executed at the server with the latest codes and the client only uses the client application, it being the web browser or a customized application.

5 SOFTWARE PLATFORMS

The project subject includes multiple different applications running on multiple different platforms. The term platform has a rather wide and inconsistent meaning in field of software development, but on this chapter it mainly consists of a certain operating system and a specific set of development tools with libraries and compiling tools.

5.1 Windows development

Developing software for Windows is rather straightforward when having the proper set of modern development tools and a slight knowledge of how to get started. After all, the development has become so easy, the class documentation has become more and more complete and in particular, the development toolchain has become so fluent that the developer will actually learn to produce the software in the middle of the working process.

5.1.1 .NET Framework

The .NET Framework consists of two main parts, the common language runtime (CLR) and the .NET Framework class library.

.NET is a managed software platform. It means that the common language runtime works as an engine, which handles some of the more tricky features of the other familiar unmanaged languages under the hood. These include features like memory and thread management. CLR provides the developer a memory management system, familiar from Java. Developer does not need to worry about freeing the allocated memory as the CLR memory management tool, the garbage collector, handles the release and the cleanup of unreferenced memory. (Microsoft, 2014.)

CLR also provides a common type system, where basic variable types are defined by the framework. This means that when developing .NET applications

with one language and then changing to some other of the supported languages, the developer can rely that the data types will not be changed. (Microsoft, 2014.)

The .NET Framework class library provides the developer a consistent coding experience between different programming languages inside the framework and a variety of complete classes ready for use instead of the programmer having to create everything from scratch (Microsoft, 2014).

5.1.2 C# language

C# is an object-oriented programming language developed by and for Microsoft. It is a type-safe language, which enables developers to build full working software solutions running on the .NET Framework. With development tools such as Visual Studio by Microsoft, developers can construct a variety of different types of applications from Windows client to Windows service applications. (Microsoft, 2014.)

The C# syntax is easy for anyone who has experience with languages like C++ and Java. The basics of the syntax are illustrated in Figure 14.

```
public void myFirstMethod(int myParam)
{
        for (int i = 0; i < 10; i++)
        {
                //doWork
        }
}
```

FIGURE 14. C# code snippet

Like most of the object-oriented programming languages, C# has specialties, which make it more suitable in some situations than other, mostly in Windows application development. For example, there are the “Delegate methods” for encapsulating method signatures and the “Properties” for wrapping private parameters for public use (Figure 15) and so greatly reducing the amount of code required for such operations.

C++:

```
private:
int myParam;

public:
int getMyParam()
{
        return myParam;
}
```

C#

```
public int myParam { get; set; } //has the private member hidden from the code
```

FIGURE 15. Example of C# Properties

5.2 Android development

Developing for Android based mobile devices is incredible easy when comparing to the Symbian S60 development. Of course the processing power of the devices has increased rapidly thus allowing more flexible development platform so it is no wonder why the world has been filled with more and more imaginary mobile applications running on Android powered devices.

5.2.1 The Android Platform

Android is a Linux based open-source platform for mobile phones and other devices like miniature-PCs and even Smart-TVs. It was created by Google and the Open Handset Alliance and it is nowadays a major platform for application developers (Burnette, 2010).

Android is an open and free development platform. It is completely free to download the software development kit (SDK) and other tools from the Internet and begin the development in few minutes. It provides the developer many built-in features for utilizing the target device as efficiently as possible. Features like GPS data, HTTP requests and high quality graphics are accessible instantly

without a need for device specific coding or downloading third-party software components. (Burnette, 2010.)

One of the main advantages of the Android platform is the fact that it is highly backed up by Google, a major player in the field of software industry. This enables the existence of one more Android feature, one of the biggest application marketplaces, the Google Play store. It is an open marketplace where anyone can get an application to be distributed for free or as a paid application (Android, 2014).

5.2.2 Android as operating system

The Android operating system is built on top of the Linux kernel. It uses the underlying Linux for memory management, process management, networking and other operating system services (Burnette, 2010). Although the developer can be totally unaware of the underlying kernel, it is highly recommended that the developer takes the Linux heritage in count.

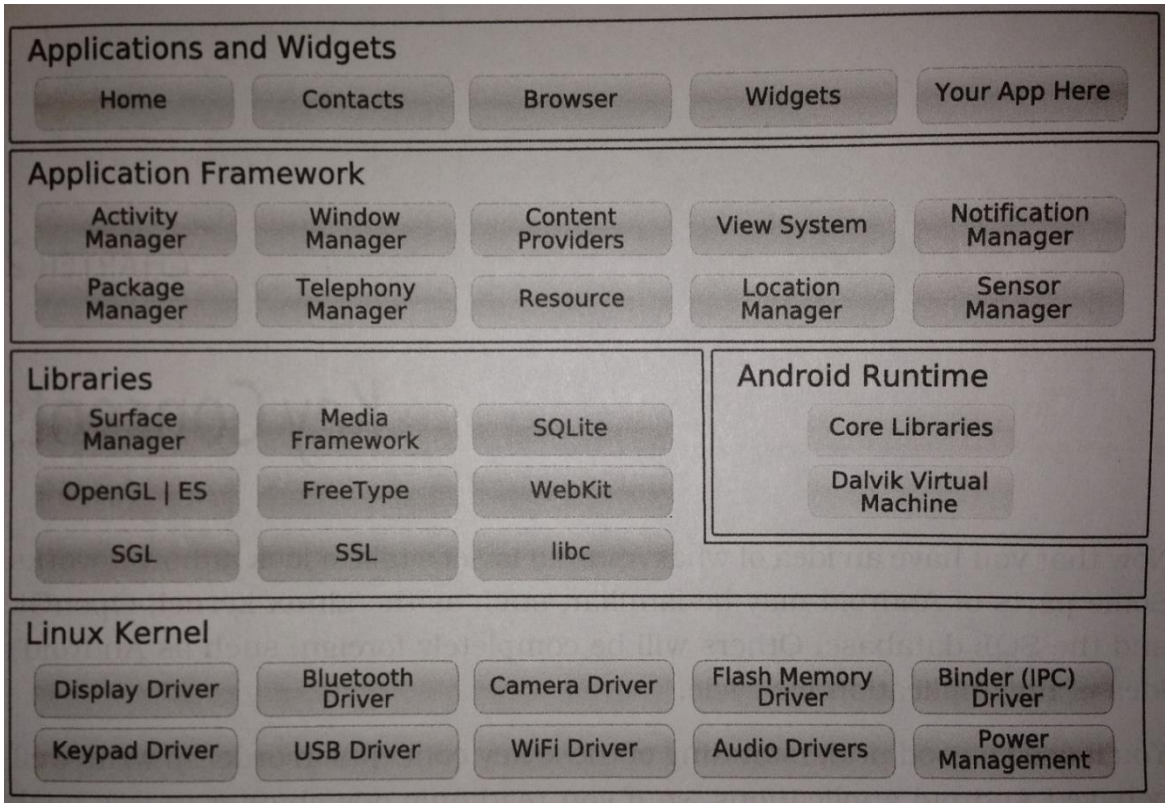

FIGURE 16. Android system architecture (Burnette, 2010)

Above the kernel layer, there are the native Android libraries. The native libraries are written with C and C++ and are compiled for a specific set of hardware. The native libraries are passive by themselves and are used by calling them from the higher architecture level. It is still possible for the more advanced developers to create their own native modules with the Google provided native development kit (NDK) and so achieve more efficient performance when done properly. *(Burnette, 2010.)*

Also on top of the kernel and on the side of the native libraries lies one of the essential parts of the Android OS, the Android Runtime including the Dalvik virtual machine with the other Java libraries. The Dalvik VM is Google's implementation of the default Java VM, optimized for the mobile device use. All Android Java code is compiled as bytecode, which is then run within the Dalvik VM. The main differences between the Java VM and the Dalvik are, that the Dalvik runs .dex files, which are converted from standard .class and .jar files. The .dex files are more compact and also more battery efficient regarding the mobile device use. Also, the core Java libraries of the Android OS differ from the Java Standard Edition (SE) and the Java Mobile Edition (ME) libraries. *(Burnette, 2010.)*

On top of the native libraries and the Android Runtime, there is the application framework layer. It provides the applications, hence the developers the actual building blocks of an Android application *(Burnette, 2010)*. The framework includes the main functionalities as the Activity Manager, the Location Manager and the Notification Manager, all familiar to every Android developer from the novice to the expert. The application layer is the final step to completing the system architecture stack.

5.3 Google App Engine

Google App Engine (GAE) is the Google's version of the "Platform as a Service" (PaaS) -service. Supported programming languages are (on 17.11.2014) Python, Java, PHP and Google-developed language Go.

Getting started with the Google App Engine is completely free. Developer can use an existing Google account or create a new one and activate the App Engine. By this, everything is set and ready for downloading the free set of tools, developing and deploying the first application. Google provides systematic starter tutorials for each language thus lowering the learning curve dramatically. (Google, 2014.)

The GAE does not only provide a traditional SQL database solution, but additionally, and as the application default, it has a unique feature called the Datastore where the data is stored as entities and indexes. This means that the Google has more control of how the data is actually stored and so is able to create more efficient and faster searches through the application datastore. Drawbacks of this solution come from the fact that the index count is limited, restricting the developer to do only certain kinds of queries as the properties (columns) used as query conditions must be indexed.

For now, the GAE supports only HTTP and HTTPS protocols. For example SSH and FTP are not supported by any way. All interactions with the Datastore are done with HTTP requests received and processed by the application itself, or by viewing and modifying the Datastore entities with the control panel. The control panel is a web site, which has all the information the developer has and needs to control the GAE application. (Salo, 2012.)

A GAE application uses instances to respond to the HTTP/HTTPS requests received. At least one instance is required for the application to work - to respond to the requests, but as a specified time span has gone with no requests, the application can go to sleep and shutdown all of the instances. At the other end of the scale, if the GAE system notices that one instance is not enough to handle the requests in short enough time, the GAE launches more instances reducing the response time but increasing the total instance time. (Salo, 2012; Google, 2014.)

Pricing in the GAE is straightforward. The more of the resources are being used, the more it will cost. At first, everything is free. As the Datastore gets more entities, request count and instance hours increase, the developer has to enable the billing system to avoid the application for freezing due the quota exceeding (Google, 2014). Some examples of the GAE pricing in Figure 17 and examples of quota in Figure 18.

Resource	Unit	Unit cost (in US $)
Instances*	Instance hours	$0.05
Outgoing Network Traffic	Gigabytes	$0.12
Incoming Network Traffic	Gigabytes	Free
Datastore Storage	Gigabytes per month	$0.18

FIGURE 17. GAE pricing (Google, 2014)

Resource or API Call	Free Quota
Frontend Instances (Automatic Scaling Modules)	28 free instance-hours per day

Resource	Free Default Limit		Billing Enabled Default Limit	
	Daily Limit	Maximum Rate	Daily Limit	Maximum Rate
Outgoing Bandwidth (billable, includes HTTPS)	1 GB	56 MB/minute	1 GB free; 14,400 GB maximum	10 GB/minute
Incoming Bandwidth (includes HTTPS)	1 GB; 14,400 GB maximum	56 MB/minute	None	None

Resource	Free Default Daily Limit	Billing Enabled Default Limit
Stored Data (billable)	1 GB *	1 GB free; no maximum
Number of Indexes	200 *	200
Write Operations	50,000	Unlimited
Read Operations	50,000	Unlimited
Small Operations	50,000	Unlimited

FIGURE 18. GAE quotas (Google, 2014)

6 PROCESS MODELS

When designing and developing a software, it can be rather dangerous to skip all the work before the actual coding process and hop right to the compiler. For easing the proper software planning, there are some process models developed.

The software process can be defined as a collection of work activities and tasks that are performed one after one (Pressman, 2010). A generic process model consists of five different activities:

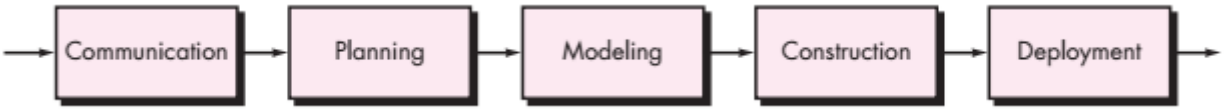

FIGURE 19. Linear process flow (Pressman, 2010)

Communication: Interviewing the customer, finding the requirements. This phase should be more or less clear also for non-software developers, as it is the phase when the wishes of the future end-user are converted into technical possibilities.

Planning: Resource estimations, scheduling. For example, project managers or team leaders schedule the estimated work hours for the requested project.

Modeling: Planning, architectural planning. For example, a software developer or a dedicated architecture developer does the modular level planning. This could consists of class diagrams of different modules and the interfaces between them.

Construction: Coding process, testing. A software developer does the implementation with all the requested features. The coding process can have multiple intermediate releases for the test team to begin testing.

Deployment: Release, support, maintenance. The software is released, the technical support team will take responsibility of responding to the customer feedback and the development team begins the maintenance, maybe by

developing and releasing hotfixes or patches for bugs, which have not been found during the actual testing process. (Pressman, 2010.)

When these elements are organized as activities which happen one after one, it is called the process flow. It describes the relation of the process activities to each other in respect of organizing and time. There are several types of process flows specified. (Pressman, 2010.)

Process flow can be linear (Figure 19), when each activity happens only once, beginning from communication and ending to the deployment phase.

Iterative process flow repeats one or more activities before proceeding into the next phase.

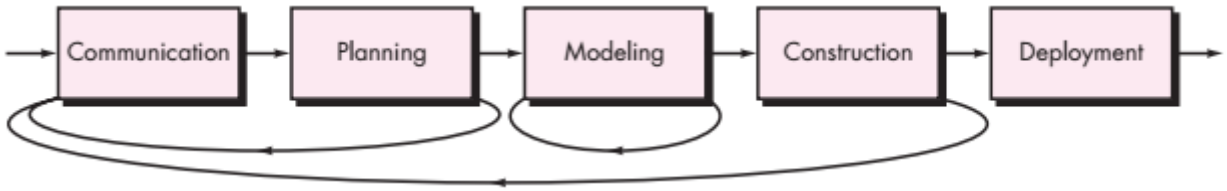

FIGURE 20. Iterative process flow (Pressman, 2010)

In the evolutionary process flow the activities are processed in a circular form, when each round through all five activities leads to a more complete software.

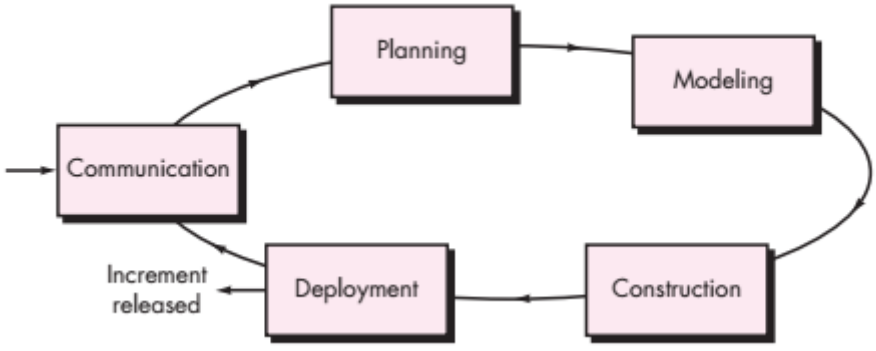

FIGURE 21. Evolutionary process flow (Pressman, 2010)

The parallel process flow prefers working with two or more activities simultaneously, when different stages of two separate processes might overlap.

For example when having to make the planning for some process regarding the construction phase of some other project *(Pressman, 2010)*.

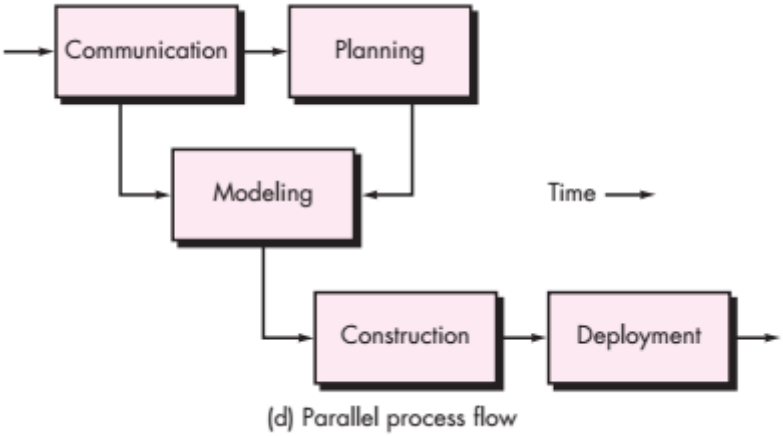

FIGURE 22. Parallel process flow (Pressman, 2010)

6.1 Waterfall model

When the problem or the need of the software and the requirements are well known and the process workflow is reasonably linear from communication to deployment, the process model can be called with the name Waterfall model. *(Pressman, 2010.)*

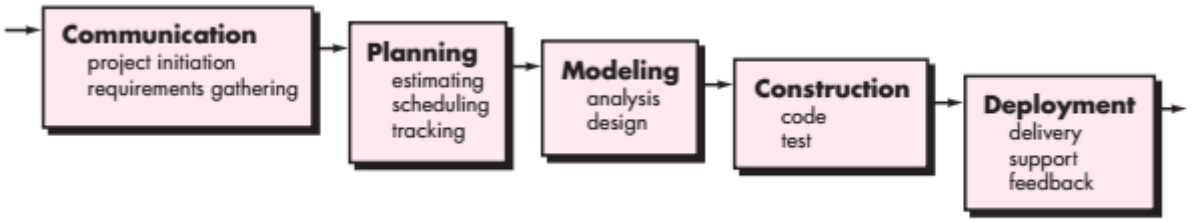

FIGURE 23. Waterfall model (Pressman, 2010)

The Waterfall model is a straightforward and strongly progressive software development process model. The development begins in a certain manner, usually by the interview of the customer, which being the communication phase. All phases are processed in specified order finally achieving the deployment phase, the actual release of the software. *(Pressman, 2010.)*

In the case of actual software development, the Waterfall model can be too restricting while it does not directly allow iterations back to the planning or even to the communication phase. As it is assumed and a well-known fact, the feature list has a tendency to be very alive during the whole lifecycle of the process. *(Pressman, 2010.)*

By the end, the Waterfall model has so many drawback that it is all too clumsy for the modern software development. It does not allow iteration rounds for rethinking the design even when the customer wants to add features during the process. Also, the actual software is coded in the very late phase of the process so the testing cannot start until the first test release is built. *(Pressman, 2010.)*

6.2 Evolutionary process models

One answer to the unwanted static behavior of the Waterfall model is to follow the more free and evolutionary model. The Prototyping model targets to get the first version of the software to the customer for inspection as soon as possible *(Pressman, 2010)*.

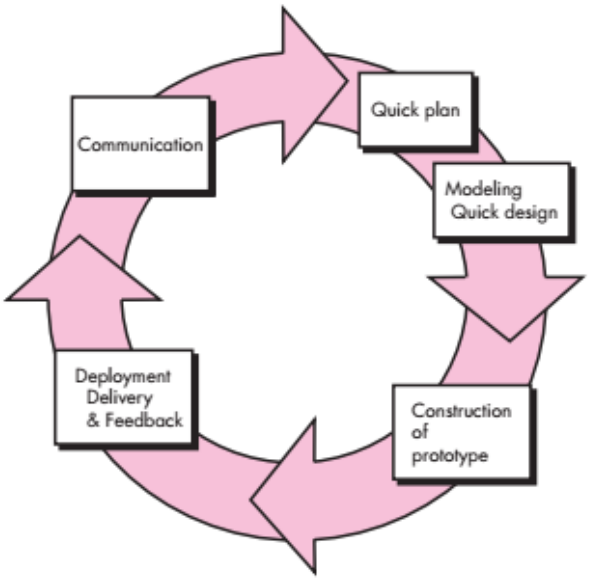

FIGURE 24. Prototyping (Pressman, 2010)

Ideally, the Prototyping process model gives the customer and the developer both a clear view where the process is going and what features could be added. The process starts with a more general view from the customer part, and it will evolve as the developer part makes progress. *(Pressman, 2010.)*

As flexible as it seems, problems with the Prototyping model raises from aspects that the customer uses an unfinished version many times and it can affect the overall feeling of quality. Although every participant is very aware about the state of the process, the customer might be expecting more reliable software than the developer is providing at that moment. *(Pressman, 2010.)*

7 LICENSE MANAGEMENT SYSTEM

The subject of the project was to generate a new flexible way of generating and managing software licenses for Nemo mobile device applications. The end user group of the application would be the logistics department of the company as well as the technical support team, both having the task of creating, deleting and updating the licenses on devices all around the world.

The system consists of several separate applications running on different platforms and operating systems. There are the customer devices with the Android operating system, the client application running on Windows desktops and laptops, which manipulates the system databases through the server which is a Windows service application running on the Windows server operating system. The final component of this rather massive system is the cloud application, utilizing the performance and capabilities of the Google Cloud platform.

While having the technological capabilities already available on the Symbian era of the Nemo products, this kind of approach had not been considered at the research and development side of the company and the logistics team could not even wish for this kind of system. After all, it is to be mentioned that the license installing to the Symbian devices was very frustrating, having to connect the device with a USB cable to a desktop computer equipped with the required software, such as the device drivers and the former license management tool.

When the initial developer of this system presented the idea for the potential end users, it was almost impossible for them to believe how much easier the license management could get when the new system was running so the first impression was rather suspicious denial. As time flew by and the first demonstrations were held, the atmosphere got more and more excited, after all this would be a great improvement to the license management tools used for everyday work. No more broken cables or USB hubs and plugs and no more waiting for the device to be recognized by the desktop application and the license being installed. The license

would come from the cloud as soon as the request was reached and processed. With the enthusiastic atmosphere spreading from developers to the end users, this was one of the most interesting projects the thesis writer has taken part during the work career.

7.1 Workflow

The workflow of the project was different to all the usual software development regarding of Nemo products because the new license management system was a whole new product and it was designed to improve the work of the logistics team and the technical support team. When having the end users, or customers near, it was possible to utilize the evolutionary waterfall workflow where the software requirements could be collected by interviewing the customer beforehand and also during the development.

7.1.1 Interview

In this case the interview was one or two meetings between the development team, logistics team and the technical support team. One problem was that the customer side did not have too much requirements when having got to use the earlier, clumsier tools and at the same time having no knowledge of what could be done better. The interview was actually an advertisement conference where the development team presented the ideas to the end users.

7.1.2 Architectural design

During the designing phase, the customer side got more active. More meetings were held and the list of requirements got more specific and clear. There is no situation regarding the software development where an accurate specification would be unwanted, so the finally achieved mutual language was very welcome.

As help for finding the common speech some architectural drawings were made. A sketch of one of the first architectural designs is presented in Figure 25. The original drawings were made on a flipchart and they were unfortunately eventually lost.

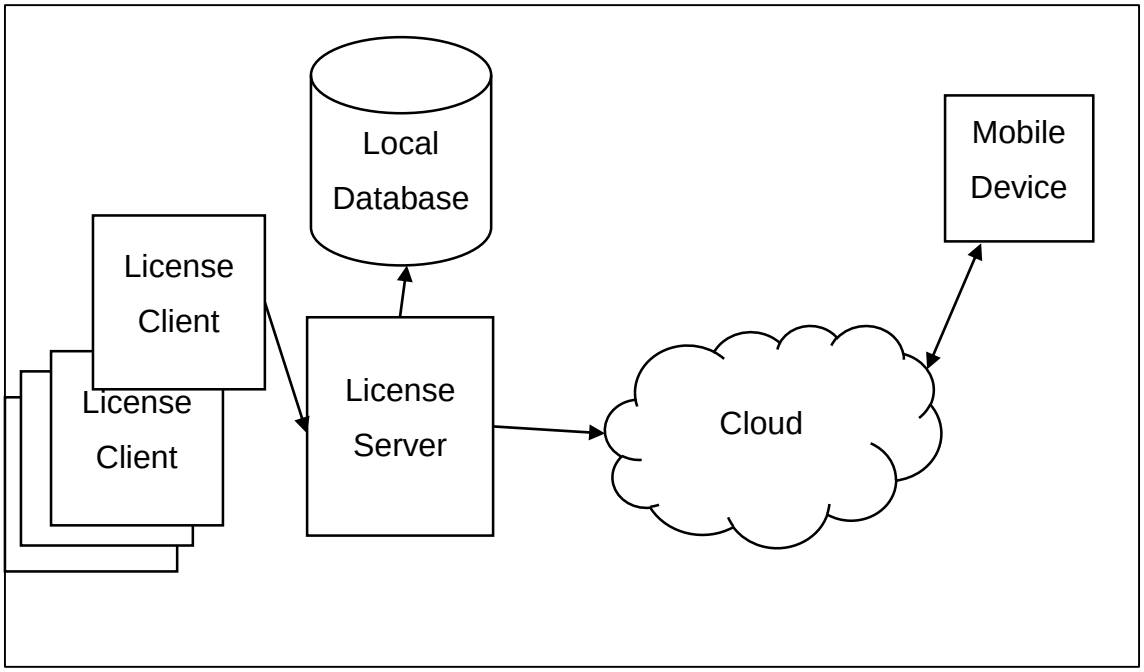

FIGURE 25. Sketch of the system architectural design

The architecture is based on the client-server model. The architecture and the actual use of the software is best presented when the workflow of the application is explained.

Describing the use case scenario of generating new license, the process is initialized by a certain logistical software running on the computer by generating a document consisting of a list of software and hardware items (Figure 26) which are needed for constructing a delivery for customer.

```
Customer No,1365
Customer PO,DEMO
Order number,205171
End customer,1365
End customer name,SomeMobileOperator
Customer name, SomeMobileOperator
Item,Ordered,Description,Serial,Delivered,EndDate
MH0700-26,Samsung Galaxy Core LTE SM-G386F,351613061207721,1,
312026-03,Nemo Handy-A Application,,1,
HA1708-23,Nemo Handy-A Pro License,,1,01.01.2040
```

FIGURE 26. Example of the item list

The request is then processed by the first part of the license management system, the License Client. The License Client processes the request to the extensive markup language (XML) form (Figure 27) and sends it as a serialized string to the License Server.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<license_request>
  <creator>matti.urhonen</creator>
  <create_date>1416230782406</create_date>
  <purchase_order />
  <order_confirmation />
  <end_customer number="" name="End Customer" />
  <customer number="" name="Customer" />
  <ticket />
  <licenses>
    <license>
      <product>Nemo Media Router</product>
      <product_version>Nemo Media Router Application</product_version>
      <options>21,22,35,46,25,23</options>
      <license_expiration />
      <ts_expiration>1447804799000</ts_expiration>
      <identifier type="IMEI">358188054852158</identifier>
      <country_codes />
    </license>
  </licenses>
</license_request>
```

FIGURE 27. Example of the request XML

The License Server processes the request in a specified manner and writes the entries into the local database, which is used as a log or a diary of what has been done. The licenses can be deleted or updated but no entries are altered or deleted from the local database whatsoever. After being properly documented to the database, the information is sent to the Google Cloud application.

The Google Cloud “Platform as a Service” (PaaS) environment, called the Google App Engine, hosts the License Cloud application. This application and the cloud data storage is the final interface between the mobile devices and the License Management system. There is a Java cloud application running on top of the Google App Engine Platform as a Service framework. The application processes HTTP requests from the License Server and does the specified task, such as adding the licenses to or removing the licenses from the data storage.

The final operator in the workflow is the mobile device in need for a license. By sending a HTTP request to the cloud application, the mobile application receives the list of available licenses as a response and is then allowed to select and download the wanted license. The license is then deactivated from the cloud application as it is saved to the local memory of the mobile device. This way it is available for recurrent use without obligatory need of Internet connection.

The full workflow of regular use case is presented in Figure 28

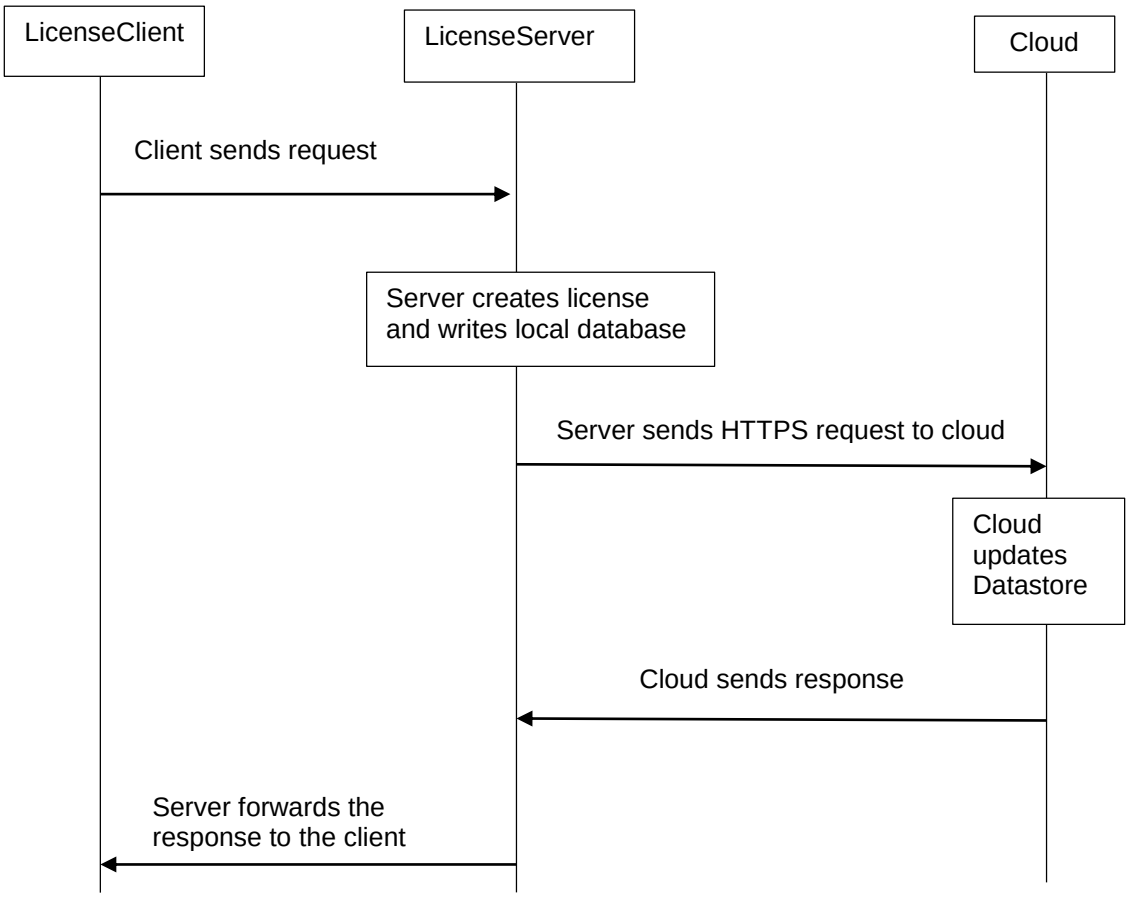

FIGURE 28. The workflow of interacting with server and cloud

7.1.3 Development

The shift from the interview and architecture designing phase to the actual development phase was a long slow but steady sliding, as new features were invented during the first demo presentations and the so called alpha testing.

At first, the development team consisted the thesis writer and one other colleague but as project went on, the whole system was assigned to the thesis writer. The development process can be divided into three separate parts, excluding the local database design and development. In the next few sections the separate components and the development processes are described.

7.2 License Client

At first the list of feature requests for the client application was unfortunately rather short. It was quite difficult to begin the development of a professional tool for professional users when the subject was very unfamiliar to the development team.

All started from some user interface sketching and having more in depth interviews with the non-technical end users, how they would like the application to look and feel like. For an engineer it is the most common situation when designing user interfaces to forget the type or the quality of the end user, as it is people who are going to use the application. Whether the end user is unfamiliar with computers, has limitations on screen size when using laptop, or maybe the end user has familiarized some other, previous software tool which would be a suitable reference for the new one. All these aspects are good to notice and make the best out of all the compromises needed to be done to satisfy as many end users as possible.

Other component of the License Engine is the network module. The network module is the interface between the License Client and the License Server, as it creates an unsecured socket connection between the two applications. In fact, the License Client can be divided as three separate operators, the user input area, the networker module and the worker. The worker handles the user input and creates the required data in specific form. The networker does the communication to the server side. The network module also has to be the most robust part of the application to ensure that there will be no unhandled, or even more critical, unnoticed errors. The error handling must always be taken seriously in software development, but especially when the product is being used as a professional tool.

The error handling is done quite a harsh way, which does not let the user to continue when even a slightest error occurs, but it guides the user to contact the administrative personnel immediately. Also, the whole application runs in a single

thread to avoid any kinds of concurrency bugs. These kinds of errors can occur very rarely, but when happening, the results could be critical and unnoticed. Figure 29 illustrates the modules and the call sequence between the modules.

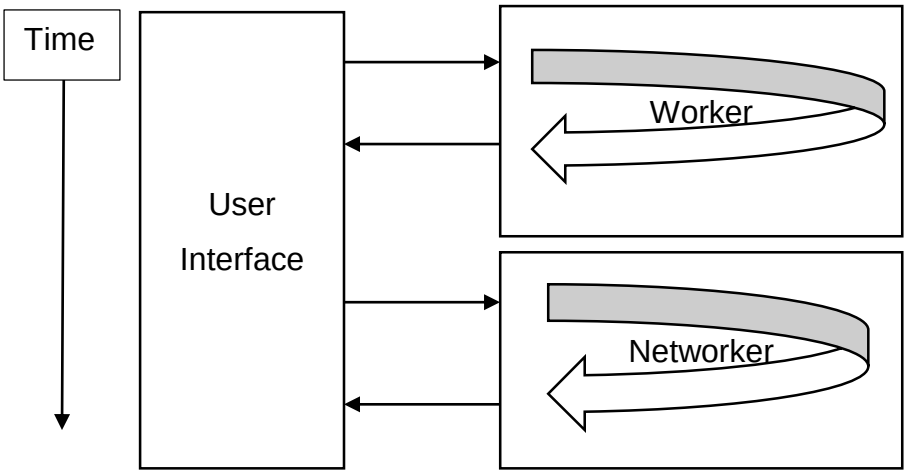

FIGURE 29. The License Client modules

The License Client has a few different user modes depending on the authorization level of the user. The two main categories are the logistics user and the technical support user. The main difference between these are the outlook of the user interface, as it is built as dynamic as needed to ensure that there is no useless features for either group shown.

7.3 License Server

The license server is a Windows service application running on the Windows Server operating system. By this setup, it is ensured as much as possible that the server application will be running all the time and is accessible whenever it is needed. After all, it is the core of all the communications in the system between the License Client and the License Cloud and any downtime could mean a halt in the whole logistics department.

The server development was very straight forward. It was not about the feature requests but more about designing a robust interface between the client application and the cloud. The feature of logging all the information history, the

so-called diary, was also to be done by the server. With all the features combined with the aspect of being a Windows service application, it is designed and more or less successfully implemented as a silent worker which requires little or no maintenance at all.

The License Server can be divided into four separate parts. There is the networker interface which acts as a spouse for the networker of the client application, then there is the database writer or the diary module, the license engine and last but not least, the cloud interface.

The networker module of the server differs from the client side much as it is the pure receiver part and has to be able to communicate with multiple clients at the same time. Every single request from any client launches a new instance of connection handler at the server side and it will have a dedicated instance of the database interface and also of the license engine interface. This can be done when there is no shared data accessed for write operations in the server side but only the database, which then has the database engine driver to ensure that the requests are queued and are not interfering with each other.

The database writer acts as a log writer for keeping the history of all actions done regarding creating or deleting licenses. In fact, the database writer can be more described as a database interface as it encapsulates all the SQL actions done to the local database. Each addition or removal request from the client eventually produces one or more database actions.

The license engine is the cryptographic part of the application where the license data is created. The input parameters transferred all the way from the client computer to the server networker module are given to the license engine and it uses the RSA and the AES algorithms to create a secure package of information which can be transferred to the cloud environment without a worry of being hacked on the way.

The cloud module of the server acts a bit like the networker, the networker can be thought as the input and the cloud interface is the output. The cloud module sends HTTPS requests to the Internet, addressed for the license application running on the Google App Engine. As the cloud application receives the request, it immediately processes the request and sends an appropriate response for the cloud module of the License Server to be inspected for possible errors. As being earlier said, the whole system depends on the reliability of the network connection and the ability of the application to catch all the errors.

7.4 Cloud

The development of the cloud application was anything but easy. The whole concept of cloud programming was new and strange at the beginning of the project and it may be said that the tools provided were not the most enjoyable to use. Still, after a while the first version of the application was ready and it was a kind of a beginning of a new era regarding the way of thinking of how to provide certain services for the mobile device customers.

The cloud application is a Java application running on top of “Platform as a Service” -service Google App Engine. The cloud application can be divided into two separate parts, the mobile device interface and the license server interface.

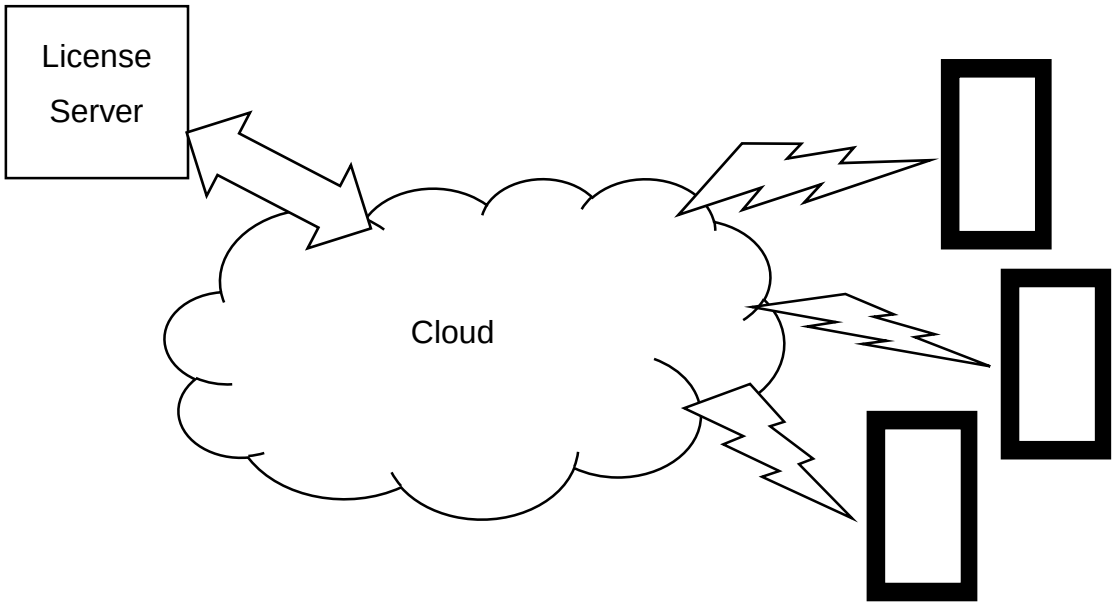

FIGURE 30. Cloud communicating with mobile device and the server

In figure 30 there are the two interfaces illustrated. At the other end, the mobile device is requesting for a list of licenses or a new license from the cloud application. The License Cloud application processes the request by assessing enough instances and computing power to be able to handle all the requests without a delay and sends the requested information from the data storage to the requestor as a response.

On the other side there is the one and only server communicating with the cloud. The server side is the only one, which can manipulate the data storage, add or remove items. All the actions are done by the cloud application by the instructions given in form of a request from the server.

7.5 Testing and maintenance

The system maintenance is an ongoing process with constant updates and bug fixes. This is the kind of project which is truly tested only by the end users and it is essentially not a good aspect in the field of software development as some of the bugs might be very critical and demand an urgent fix. By now, there have

been several updates to each part of the system consisting of a few hotfixes with only a few hours or even less time to be done.

There was no final testing in its true form even before the first release, but the release process included a period of time where the development team was on continuous alert as the logistics team began to experiment with the new system. The same routine repeats itself still these days after every major update, mostly done to the server side.

8 CONCLUSION AND DISCUSSION

In my point of view, the project has been a success. For backing up my opinion, there have been a lot of thank-yous from the departments who use the system as a tool in their everyday work. After all, the result of the project has been affecting the efficiency of the whole global logistics team, which was having problems delivering the products with the older high-maintenance and very time consuming license generator tool.

As a result, all the predefined requirements were achieved. The new license management tool is more flexible and, above all, very fast compared to the previous licensing tool. The new tool has been in everyday use for about a year now and it has established its place in the workflow of the logistics and the technical support teams. Also, the secondary objective, to research cloud service possibilities, was a success. We gained a lot of important and useful information and experiences regarding the cloud services. Especially the Google App Engine has been a delightful new development platform in the web application development genre, and more importantly, it has been robust and reliable enough for our use so far.

When thinking back the start of the project, it was a very special occasion as the system was being developed from scratch with all the possibilities ahead. And on top of that, there was a decent possibility to plan the accurate specifications and even the architecture models to help the development process and to keep the system functional and clean. This kind of working method is unfortunately too rare in our task related work with the main products, where the new features are built on top of the older features when a customer happens to need one.

The project was also a great learning experience as it consisted of multiple new categories from cloud applications to Android software development. The motivation during the project was high as it was basically in my responsibility alone at the end and the subject was not too hard to handle. In fact, there were no specific moments or periods of time when the project would be delayed for

technical reasons or difficulties. The main line was always known and the deadlines also kept their places.

Of course, there are few drawbacks. It has been noticed that the current system might not be flexible enough to maintain all the changes the future feature requests could demand, so there is a risk of a great refactoring for the code base. But these kind of changes happen in the software development life and they must be handled properly.

8.1 Cloud services

This was by far the most interesting part of the project. The development speed of the web applications enables creating something fresh and new, and it surely gave some motivation for some recreational code development.

One great example of possible Platform as a Service project is an online gaming system. With the development tools provided, and also inside the limits they set, the developer can concentrate in building the most entertaining game and let the cloud service provider to ensure that their capabilities match the promised quality. In the gaming industry it may even be impossible to know how rapidly the new game will attract players. In case where the system is underestimated, possible clients are lost and that directly means a lost profit.

In my opinion cloud services are here to stay. When considering from the consumer point of view, the cloud services are an amazing way of storing data in one rather secure place, which is accessible from any Internet enabled device. As from the commercial point of view, companies are eagerly trying to find new ways of reducing their running costs. Cloud services, cloud computing power and the Platform as a Service applications provide an efficient way of getting rid of the hardware equipment which otherwise would constantly need maintenance and possible upgrading.

Platform as a Service cloud service model gives the user the flexibility of creating their own applications with very little restrictions. As removing the logistic aspect of keeping the client software up to date and giving the scaling resources for ensuring the computing capability it is most presumably the platform choice of many future application developers.

The main question and challenge for the cloud service providers is the demand for data security and especially the demand for robust system with no down time. When the level where these aspects are no longer an issue is achieved, there is no limit what can be done with cloud applications.

REFERENCES

Salomon, D. (2003). Data Privacy and Security: Encryption and Information Hiding

Daemen J., Rijmen V. (2002). The Design of Rijndael: AES - The Advanced Encryption Standard

Stallings, W. (1999). Cryptography and Network security, principles and practice

TechTarget (2014). What is Rijndael? Retrieved November 2014, from http://searchsecurity.techtarget.com: http://searchsecurity.techtarget.com/definition/Rijndael

Wong, C. (2000). HTTP Pocket Reference: Hypertext Transfer Protocol

SHA-1 online (2014). Retrieved November 2014 form, http://www.sha1-online.com/

TechTarget (2014). What is RSA algorithm (Rivest-Shamir-Adleman). Retrieved November 2014, from http://searchsecurity.techtarget.com: http://searchsecurity.techtarget.com/definition/RSA

Microsoft (2009). Chapter 1: What is Software Architecture? Retrieved November 2014 from http://msdn.microsoft.com: http://msdn.microsoft.com/en-us/library/ee658098.aspx

Oracle (2001). Anatomy of the Client/Server Model. Retrieved November 2014 from http://docs.oracle.com: http://docs.oracle.com/cd/E13203_01/tuxedo/tux80/atmi/intbas3.htm

Pressman, R. (2010). Software Engineering: A Practitioner’s Approach.

Garlan, D. & Shaw, M. (1994). An Introduction to Software Architecture. Retrieved November 2014 from http://www.cs.cmu.edu: http://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf

Google (2014). Google App Engine – Platform as a Service. Retrieved November 2014 from https://cloud.google.com: https://cloud.google.com/products/app-engine/

Salo, I. (2012). Hyötyä pilvipalveluista. Docendo 2012.

Microsoft (2014). Getting Started with the .NET Framework. Retrieved November 2014 from http://msdn.microsoft.com: http://msdn.microsoft.com/en-us/library/hh425099(v=vs.110).aspx

Microsoft (2014). Introduction to the C# Language and the .NET Framework. Retrieved November 2014 from http://msdn.microsoft.com: http://msdn.microsoft.com/en-us/library/z1zx9t92.aspx

Microsoft (2014). Common Type System. Retrieved November 2014 from http://msdn.microsoft.com: http://msdn.microsoft.com/en-us/library/zcx1eb1e(v=vs.110).aspx

Microsoft (2014) .NET Framework Class Library Overview. Retrieved November 2014 from http://msdn.microsoft.com: http://msdn.microsoft.com/en-us/library/hfa3fa08(v=vs.110).aspx

Burnette, E. (2010). Hello, Android. Introducing Google’s Mobile Development Platform, The Pragmatic Bookshelf 2010

Android (2014). The world's most popular mobile platform. Retrieved November 2014 from https://developer.android.com: https://developer.android.com/about/index.html

Google (2014). Google App Engine: Platform as a Service. Retrieved November 2014 from https://cloud.google.com/: https://cloud.google.com/appengine/docs