An Example of the 4-Tier Security Architecture on Transmission Electronic Documents

Siniša Vujčić,
Depart. of IP/MPLS network administration and maintenance
M:tel – Telekom Srpske
Banjaluka, R. Srpska, B&H,
sinisa.vujcic@mtel.ba

Abstract—**In this paper is described an example of 4-layer cryptography architecture aimed for strong cryptography protection of electronic documents. Brief introduction, technical background and development environment are given at the beginning of paper. Main goal was to develop client/server cryptographic software processor that have implemented: SSL/TLS client to server connection, digital signing, digital envelope and data scrambling crypto protection. It was given design proposal and solution description. Finally, system was tested and results are presented. Conclusion summary and possible future development are given at the end of paper.**

Keywords-cryptography, encryption, digital evelope, data scrambling, smartcard

I. Introduction

A Document Management System (DMS) [1] is the organizational process of all documents from the creation, storage and organization, search and access, to the destruction at the end of lifecycle. An Electronic Document Management System (EDMS) stores electronic versions of documents, usually, scanned copies of paper documents and documents that have already been produced electronically. Typical topology of the EDMS is shown on Fig.1. Documents are usually organized into a file/folder system, which allows users fast retrieval through browsing or indexing systems. [2]

A good electronic document management system is one of the best ways to streamline business processes and increase productivity. As we can suppose the Document security is vital in many document management applications. The Work described in this paper is focused and dedicated on that very important segment of EDMS. We could define next security problems:

- Secrecy - Keeping information private out of unauthorized parties.
- Authentication-Proving users identity, before revealing Documents.
- Non-repudiation - proving that a message was sent by users
- Integrity - Proving that a message wasn't modified during transmission.

Main goal of described work was to propose, plan and design one solution of four-tier security system capable to well enough secure documents in EDMS system. It was planed that system have included security layers below:

- Transport layer security
- Digital signature
- Digital envelope
- Data scrambling (data masking, obfuscation)

These features should ensure next important requirements.

- Confidentiality - data accessed and read only by authorized parties
- Integrity – data modification by authorized parties
- Availability – data available to authorized parties

It was planed that system should be designed as two-tier client/server software system capable to be hosted as Stand-alone application or as software module in bigger software system.

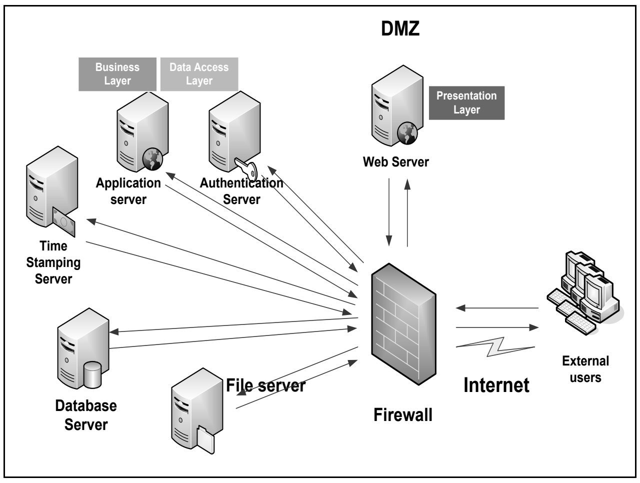

Figure 1. Typical architecture of the EDMS.

II. TECHNOLOGY BACKGROUND AND CRYPTOGRAPHY TECHNIQUE

This chapter describes brief introductory regarding the cryptography methods implemented and technology used to design proposed system. Under term File Cryptography protection we can assume the theory and implementation of techniques for secure communication between communication peers in the presence of third parties (adversaries). Cryptography techniques are differentiated based on the layer of OSI (The Open Systems Interconnection) model on which are implemented. The TLS (Transport layer security) and its predecessor SSL (Secured socked layer) are cryptographic protocols of the fourth layer (Transport) of the OSI model. It assumes asymmetric cryptography for key exchange, symmetric encryption for confidentiality, and message authentication codes for message integrity. A valid sender digital signature gives a recipient reason to believe that a known sender created the message, and that it was not altered during transmission.

Digital envelope is a type of security that uses two layers of encryption to protect a message. First, the message itself, is encoded using symmetric encryption, and then the key to decode the message is encrypted using public-key encryption. This technique overcomes one of the problems of public-key encryption, due to low processing speed of asymmetric encryption.

Data scrambling, (data masking) is a security measure designed to protect confidential and sensitive data from both internal and external threats by masking sensitive data to prevent the risk of exposing it to unauthorized users. There are few different ways in which it can be masked. It this case, the way is transposition block of document and rearrange this block according the generated pseudo-random sequence.

So, designer goal was to design software system that has all four techniques implemented in. Next chapter describes the way of implementation and software architecture.

III. DESIGN

In this chapter is described software development kit and hardware platform used to develop and test proposed system.

Both subsystem, client and server, were developed using Java programming language. All source code was written and tested using Oracle JDeveloper 11g, as a code designer. Java JDK 1.6.31 was used as a software development kit. Two software cryptography libraries were used to upgrade crypto functionality:

- Bouncy Castle crypto libraries (BC) [3] are an open-source lightweight cryptography API for Java and C#. In this project it was used to generate digital envelope and PKCS7 format digital signature. PKCS is a group of public-key cryptography standards devised and published by RSA Security Inc, starting in the early 1990. PKCS7 is Cryptographic Message Syntax Standard used to sign and encrypt messages under a PKI and often used for single sign-on.
- J4sign [4] is an extension of the open source BC for using PKCS#11 tokens. It was used as a library for generation external signed PKCS7 digital signature.

Two hardware platforms were used as test machines:

a) Laptop model HP nx6320 based on prosessor Intel T2500 2GHz, with 1Gb RAM running MS Windows XP Professional SP2. In this context it was used as a client side machine
b) Laptop model ASUS k53u based on prosessor AMD C-50 1GHz, with 2Gb RAM, running MS Windows XP Professional SP3. In this context it was used as a server machine.
Both computers were connected with crossover UTP cable making very simple, L2 based, client/server network.

Also, it was used USB smart card reader OMNIKEY cardman 3021 as well as dedicate dll middleware file.

Both subsystems, client and server, were developed to accomplish next goal features:

- First security layer (Transport layer security), realized as a SSL/TLS security channel between communication peers. In this case it is based on bilateral (client and server) authentication. Client Certificate is located in jks file (Java key store) on server machine. Server certificate is located in smartcard or in client jks file, depending on operation mode.
- Second security layer (Digital signature), is application layer security based on digital signing and verification functionality. As a result of procedure, there is a generation of PKCS7 formatted digital signed file (document).
- Third security layer (Digital envelope), is application layer security based on asymmetric protection of the Secret Key used to generation of a cipher. In short, The Document (file) is encrypted using symmetric cryptography. Secret key is generated and used as a key for encryption. Due to leak of speed of asymmetric encryption, symmetric encryption is used to protect the payload file. Asymmetric cryptography was used only to encrypt the Key. Encrypted Document file, as well as, enclosed protected key making The Digital envelope.
- Fourth security layer (Data scrambling), is application layer security based on data hiding technique. Source document is split in n blocks, later rearranged to generate scrambled file. One new and random generated block was intentionally added to enforce security in case possible cryptanalysis attack.

Client software supposed to be hosted on client machines (work places). It is possible to run the client as Standalone application or as a module in bigger software systems (EDMS client side Graphical User Interface). Server side software has a role of application and file server. It is based on listener and file manager subsystem. It supposed to be a part of The EDMS server system, or work as a standalone application depending on type of usage. Two jks (Java Key Store) have been generated, as well as, key pairs in both of them. One jks file was stored on client machine and the other on the server machine. Self signed certificates were generated and exported. Client certificated has been imported in server jks file and server certificate has also been exported and imported into client jks. This procedure has to be obtained due to using this certificates and private keys during the cryptography processing.

Production usage is based on two modes:

- Upload mode, used in case of document transmission from client side to server side software. It is initiated in case of storage new file on the document server.
- Download mode, used in case of demand for document that had already been archived on EDMS server. It is initiated by client request for document and served by server, which encrypt requested document and send delivered it to client.

The following paragraph describes fourth security layer based on data masking technique. This layer should increase general security by applying Transposition and Detransposition process on a preprocessed document. Complete process is presented by algorithm shown on Figure 2. The main goal was to change structure of the document by scrambling (Transposition) and extend time necessary to possible encryption, and detection of successful crypto-attack. Source document for this layer is previously signed and enveloped document.

Transposition process begins with the random sequence generation. The resulted document is rearranged according to this sequence. The redundant block is randomly generated and inserted in scrambled document. Random generated sequence was encrypted by asymmetric algorithm and sent during the transmission process. This sequence was also used to detect redundant block and reassembling source document. During the Transposition process this (redundant) block was located on zero position. Process of detransposition used to ignore the redundant block. Table I. shows time estimation necessary for cryptanalysis of the sequence order by brute-force attack.

TABLE I. BRUTE-FORCE ATTACK TIME PREDICTION

No of blocks	*Max combination No*	*Time estimated {days}*
13	87178291200	1
14	1307674368000	15
15	20922789888000	242
16	355687428096000	4117
17	6402373705728000	74101
18	121645100408832000	1407929

Estimation was based on 10^6 attacks per second. In this case time required is computed as the variation number without repetition.

Client side software was designed as java application hosted on client machines, and intended to handle client request for download and upload files (documents) to EDMS server. Software design is shown by algorithm shown on Fig. 2. As it was mentioned above, client software was implemented to perform all four layer of cryptography protection. It is important to notice that client software is responsible to initiate connection to server and utilities smartcard to generate digital signature and/or generate digital envelope. Structure of processed document is shown on Fig.5

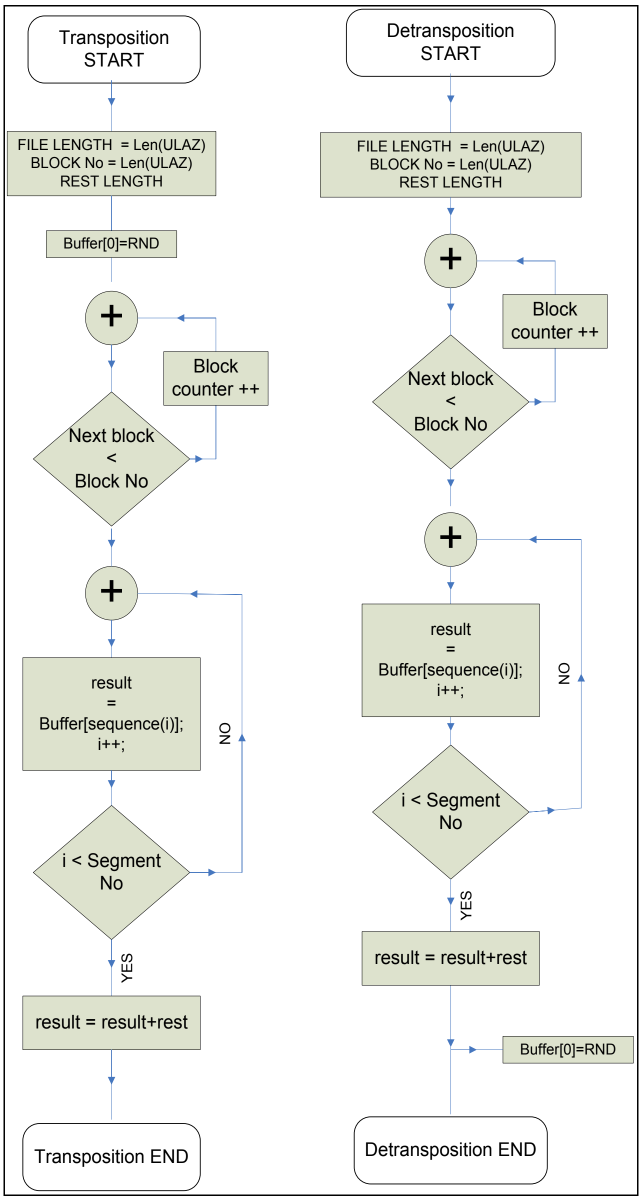

Figure 2. Document Transposition and Detransposition algorithm.

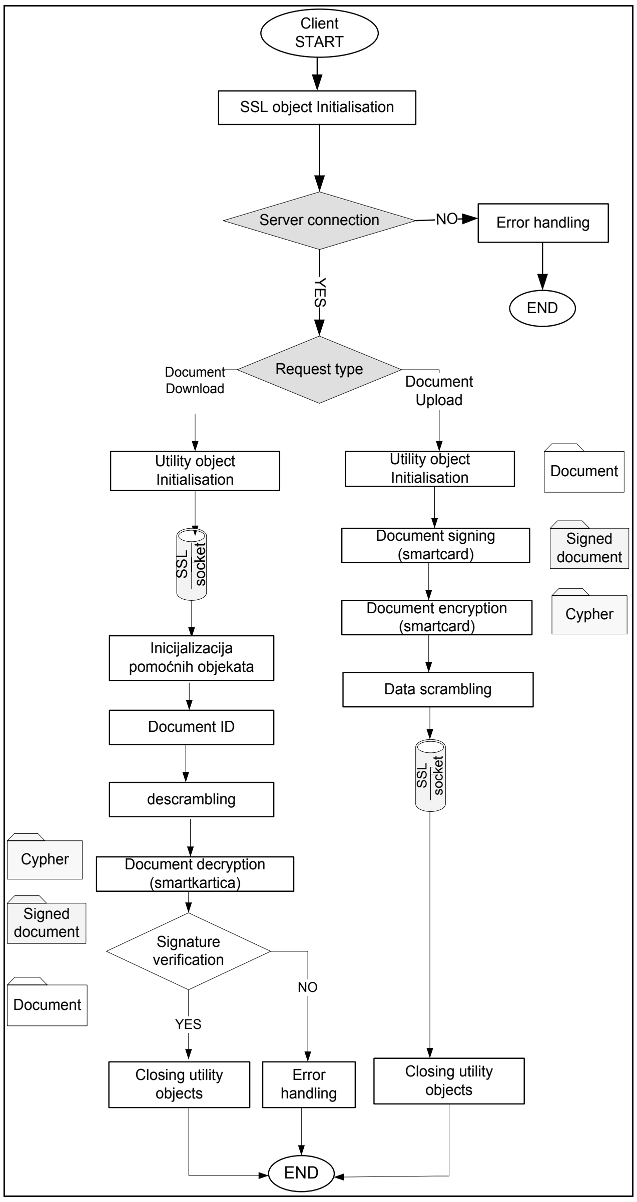

Figure 3. Client module algorithm.

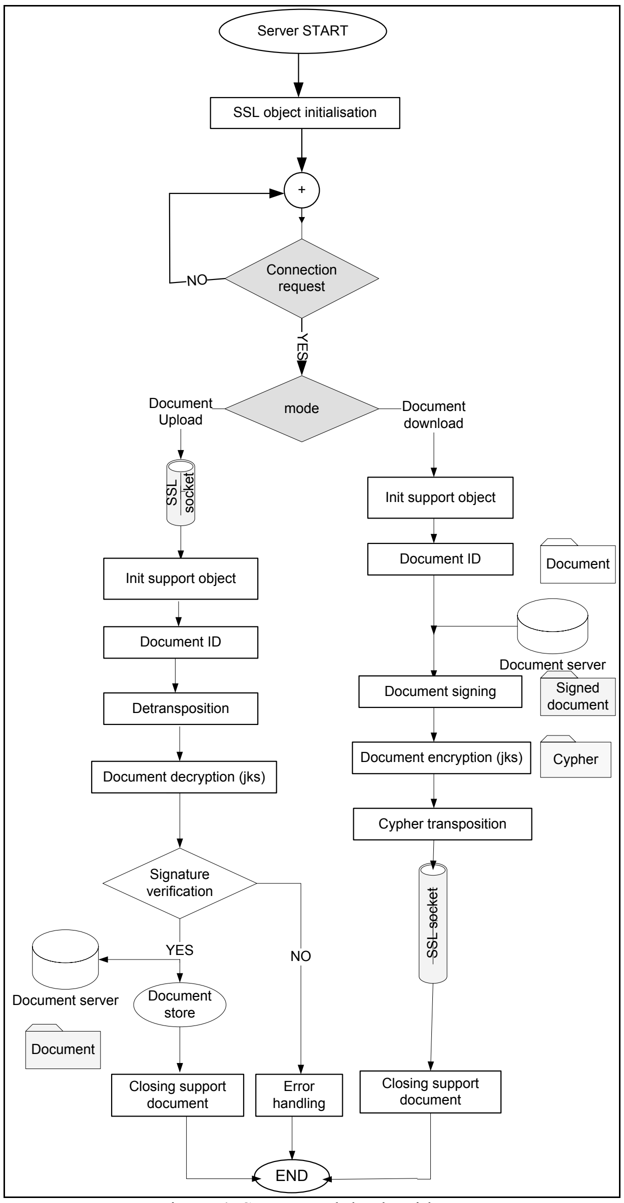

Figure 4. Server module algorithm.

Start parameter are:

- Server ip-address,
- Port number, over which connection is going to be established
- Mode, upload document or request for download.
- Name of the document requested or to be

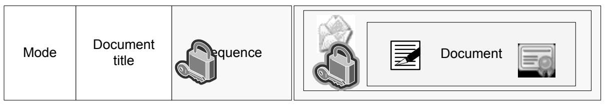

Figure 5. Structure of Encrypted Document.

- Configuration type, which includes operation mode due to source of certificates.

Server side software was designed as java application hosted on server machines, and intended to handle client request for download and upload files (documents) to EDMS. Software is capable to handle many client requests using multithreading. Software design is shown by algorithm shown on Fig. 4. Start parameter are port number on which is listener configure to listen the client requests.

IV. TESTS AND RESULTS

Test measuring have been intended to measure achieved system performance. Test lab was made by simple L2 connection of two hosts running client and server software respectively. Parameterization was based on source of the certificates and document size. All measuring have been based on approximately 50 measuring. Measured value was average value of every series. Approximately 5% of samples of every series (min. and max.) have been rejected as incorrect due to impact of background processes of the laptop operating system. The way the measuring has been performed was based on starting client software and store measured results in log file. Server software has been running for all the time of measuring. There were two System functions capable to measure time of process execution and those are: System.currentTimeMillis() and System.nanoTime(). In the Table belov have been shown main characteristics of both of them.

TABLE II. MEASURING METHODS. MAIN FEATURES AND OVERVIEW.

	Function	Precision	Time Source
1	**System.currentTimeMillis()**	1ms	System clock
2	**System.nanoTime()**	1ns	CPU counter

First method shown has a more reliable time source, but the second one is much more accurate. Two measuring methods have been compared on independent test and it was measured that difference in between them is less than 1%.

First Measures were performed in three series:

- First series assumed that client software conntact only jks archive to fetch necessary certificates and private keys
- Second series assumed that client software conntact jks archive to fetch necessary certificates but access private key stored on smartcard
- Third series assumed that client software conntact only smartcard to fetch necessary certificates and private keys.

Second measure was intended to explore impact of file size (document size) and transmission direction to processing time.

TABLE III. COMPARED RESULTS OBTAINED ON THE SAME HARDVARE PLATFORM (CLIENT AND SERVER HOSTED ON THE SAME MACHINE)

Key location	Upload [μs]	Upload Impact rate [%]	Download [μs]	Download Impact rate [%]
jks file	1319,3	38	1459,93	71
jks/smartcard	3086.4	89	1249.9	61
smart card	3466,4	100	2056,2	100

TABLE IV. COMPARED RESULTS OBTAINED ON THE SEPARATED HARDWARE PLATFORM S (CLIENT MACHINE AND SERVER MACHINE).

Key location	Upload [μs]	Upload Impact rate [%]	Download [μs]	Download Impact rate [%]
jks file	1695,7	43	1878,4	66
jks/smartcard	3508.5	89	2052	72
smart card	3943,6	100	2859,9	100

TABLE V. IMPACT OF FILE SIZE, TRANSMISSION DIRECTION TO PROCESSING TIME

Mode	Key length	Document [Mb]	Processing time [μs]	Download Impact rate [%]
download	1024	1	2052	29
upload	1024	1	3508.5	50
download	1024	5	4250.2	61
upload	1024	5	4613.4	66
upload	1024	10	5730.5	82
download	1024	10	6981.4	100

V. CONCLUSION AND FUTURE WORK

The software design and results shown in this paper are excerpt of bigger research, that also assumes cryptography performance analyze. It has been shown The Idea and practical design of 4-layer cryptography protection system with all the mainstream features (SSL, digital signature and digital envelope). Also, it was shown possibility that all the client side features can be implemented on smart card only, so there is no need to have jks file stored on client. Presented results show that there isn't significant decreasing of performance by using smart card instead of using just jks files. It is also important to note that jks file security is based on password used to encrypt the file itself. On the other hand, smart card security is based on smart card hardware, so there aren't possibility to read private key stored on it due to hardware protection. The reasons for not to use smart cards are minor comparing to benefits of better keys security. Future work might be headed to construct better algorithm for data scrambling. In this context random function, used as sequence generator, was implemented by standard java classes. Using smart card to generate random sequence might solve this functionality. Next necessary feature might be including time-stamp in processed document.

REFERENCES

[1] Jedno rješenje softverskog sistema za arhiviranje i manipulaciju dokumentima baziranog na open source softveru, Siniša Vujčić, Infoteh 2007, hotel Bistrica, Jahorina

[2] http://www.techopedia.com/definition/12769/electronic-document-management-system-edms

[3] http://www.bouncycastle.org/index.html

[4] http://j4sign.sourceforge.net/index.htm