Securing Service Access with Digital Certificates

Best Practice Document

Produced by the AMRES-led working group on security

Author: Milica Kovinić (AMRES). Contributors: Dušan Pajin, Mara Bukvić, Marko Stojaković, Ivica Barišić, Bojan Jakovljević

April 2015

Document No:	GN3-NA3-T4-AMRES-BDP-106
Version / date:	Version 2.1; April 2015
Original language	Serbian
Original title:	"*Upotrebom digitalnih sertifikata do sigurnog pristupa servisima*"
Original version / date:	Version 1.0, September 2010. First revised: April 2011
Contact:	sigurnost@amres.ac.rs

AMRES/RCUB is responsible for the contents of this document. The document was developed by the Security Topic Group organised at AMRES with the aim of implementing joint activities on developing and disseminating documents containing technical guidelines and recommendations for network services in higher education and research institutions in Serbia

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 605243, relating to the project 'Multi-Gigabit European Research and Education Network and Associated Services (GN3plus)'.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

This document promotes the adoption of digital certificates in the member institutions of the Academic Network of Serbia (AMRES) as a means of establishing secure communication channels.

In order to establish secure communication when receiving or sending data from/to a server, users must be sure that they are indeed accessing the resources they intended to access and that no one can read and/or change the data that is being sent or received. Such security is provided by the use of digital certificates in conjunction with SSL technology.

This document outlines the components of the PKI infrastructure, and also the implementation of PKI functions in the case of including AMRES in the TCS service (TERENA Certificate Service). Likewise, this document specifies various needs for PKI in NREN, which require various types of digital certificates, while special attention has been given to the use of the PKI infrastructure and digital certificates in combination with SSL technology for the purpose of the mutual authentication of services and their users.

This document explains the procedure for obtaining a server certificate – key generation, the creation of certificates and the preparation and submission of the request for signing a server certificate. The final part of the document contains instructions for installing digital certificates on Linux servers.

Rezime (Serbian version of the summary)

Dokument promoviše usvajanje digitalnih sertifikata u institucijama članicama Akademske mreže Republike Srbije (AMRES) kao načina za uspostavljanje sigurnih kanala komunikacije.

Da bi korisnici prilikom preuzimanja ili slanja podataka na neki server imali zaštićenu komunikaciju, moraju biti sigurni da su zaista pristupili onom serveru kojem su imali nameru da pristupe i da niko ne može pročitati i/ili promeniti podatke koji se šalju ili primaju. Upotreba digitalnih sertifikata u kombinaciji sa SSL tehnologijom omogućava pomenutu sigurnost.

Opisane su komponente PKI (*Public Key Infrastructure*) infrastrukture, ali i način realizacije PKI funkcija na primeru uključivanja AMRES-a u TCS (*TERENA Certificate Service*) servis. Navedene su i različite potrebe za korišćenjem PKI u nacionalnim istraživačkim i edukativnim mrežama odnosno nacionalnim akademskim mrežama (*National Research and Education Network, NREN*), koje zahtevaju različite tipove digitalnih sertifikata, ali je posebna pažnja posvećena korišćenju PKI infrastrukture, odnosno digitalnih sertifikata u kombinaciji sa SSL tehnologijom u svrhu međusobne autentifikacije servisa i njihovih korisnika.

U dokumentu je objašnjen postupak pribavljanja serverskog sertifikata – generisanje ključa, formiranje sertifikata, priprema za/i podnošenje zahteva za potpisivanje serverskog sertifikata. U završnom delu dokumenta nalaze se uputstva za instalaciju digitalnih sertifikata na Linux serverima.

1 Introduction

As a result of the development of electronic communication, confidential information is exchanged on a very frequent basis. In order to receive or send sensitive data (e.g. usernames) when accessing web content, e-mail or RADIUS servers, users must be certain that they have accessed the right server, that the communication with the server is secure and that no one can intercept/read and/or change the data. The use of SSL technology on servers (HTTPS, POP3S, etc.) provides the necessary security, though the servers need to have the appropriate digital certificates.

AMRES enables the issuance of server and personal SSL certificates in cooperation with TERENA, through the TERENA Certificate Service (TCS). The service is free of charge for all the member institutions of AMRES.

The purpose of this document is to promote the use of server and personal certificates that allow the establishment of more secure communication channels. The document is intended for the IT administrators of the member institutions of AMRES.

The first part of the document provides definitions of terms and expressions, cryptographic systems, protocols, and techniques concerning the use of digital certificates and the Public Key Infrastructure. This part also describes the format and types of digital certificates for different purposes in the Academic Network.

The second part of the document addresses the procedure for obtaining a server certificate – key generation, the creation of certificates and the preparation and submission of the request for signing a server certificate. The final part of the document contains instructions for installing digital certificates on Linux servers.

2 Basic Terms Connected to Secure Communication and PKI

This document uses technical security terms related to the use of the Public Key Infrastructure to enable secure communication in computer networks. For the sake of a better understanding of this document, the definitions of the most important terms are singled out and provided below.

Computer security is based on confidentiality, integrity and the availability of data, while secure communication implies the preservation of confidentiality and integrity and authentication with non-repudiation.

- **Authentication** is the process of establishing the identity of the end users in communication (e.g. using digital signatures).
- The **Integrity** of the data guarantees that there has been no change to the data or content of a message on its way from the source to the destination, most often by generating a hash of the message (a sequence of fixed-length bits) that is sent along with the message. Upon receipt of the message, a hash is generated based on the same algorithm used when the message was sent, which is then compared with the hash received together with the message. If these two values do not coincide, this means there has been a change to the original content of the message.
- The **Confidentiality** of data ensures that the data or the content of a message is only available to the intended recipients, which is achieved by way of encryption.
- **Non-repudiation** prevents a person who sent a message from subsequently claiming that he/she did not send the message. When a sender digitally signs a message, it is known that he/she has used his/her private key to make a digital signature. When it is only the sender who has access to his/her private key, this means that only the sender could send the relevant digitally signed message.

Cryptographic keys include:

- **Asymmetric Key Pair**, which contains a private and a public key, as a mathematical pair that is used for the operation of an asymmetric cryptographic algorithm, such as the RSA algorithm. A message encrypted using the public key can only be read by the private key owner, while a message encrypted by the private key can be read by everyone who knows the public key.

- **Private Key** – mathematical data that can be used for creating an electronic signature or decrypting a document encrypted using an asymmetric cryptographic algorithm. The private key is confidential and available only to its owner.
- **Public Key** – mathematical data that can be published (most often in the form of an X.509v3 electronic certificate). This is used for verifying an electronic signature created using the appropriate private key, which is a mathematical pair with a given public key, as well as for encrypting data for a user that has the appropriate private key.

- **Shared Secret Key** – mathematical data used for encrypting and decrypting a document encrypted using the symmetric cryptographic algorithm. The terms 'symmetric key' or just 'secret key' are also used.

Public Key Infrastructure (PKI) contains the hardware, software, policies and procedures needed to manage, generate, store and distribute cryptographic keys and digital certificates. PKI is a secure infrastructure whose services are implemented using an encryption system concept involving asymmetric algorithms. The following definitions are used in relation to the PKI infrastructure:

- **Certification Authority (CA)** – a legal entity that issues digital certificates.
- **Registration Authority (RA)** – an entity responsible for the initial identification and verification of data on a certificate user/owner, which does not issue or sign certificates. The RA is delegated by the CA to perform appropriate actions concerning verifying the identity of end users.
- **Digital certificate** – an electronic document confirming that the public key belongs to a certain entity (RFC 5280). The term 'electronic certificate' is also used.
- **CA certificate** – the certificate of the CA itself. This confirms that the CA is precisely the CA it claims to be. It can be self-signed (when the CA is also the Root CA) or issued (digitally signed) by another Certificate Authority – certificate issuer.
- **Certificate chain** – an ordered sequence of certificates that is processed together with the public key of the initial object in the chain in order to verify the public key of the last object of the chain. A hierarchical chain of trust is formed, where one digital certificate confirms the authenticity of the previous digital certificate.
- **Certificate Policy (CP)** – a designated set of rules that define the applicability of a certificate to an environment and/or a class of applications with common security requirements.
- **Certificate Practice Statement (CPS)** – public practice rules and procedures applied by the Certification Authority in the process of issuing certificates.
- **Third party** – refers to a certificate recipient who accepts the certificate and verifies its validity. In addition, the recipient verifies the digital signature of electronic documents signed by the certificate using the public key of the certificate owner, which is included in the certificate itself. In order to verify the validity of a digital certificate, the third party needs to verify the status of revocation of the certificate by verifying the appropriate CRL list.

- **Signatory** – an entity that has a means for electronic signing and performs electronic signing in its own name. It can be an individual or an entity within a legal person to whom a certificate has been issued.
- **User (end entity, certificate owner)** – an individual or an entity within a legal person to whom a digital certificate is issued. Individuals are end users, while legal persons are institutions that request digital certificates for their entities, most often for their servers or services.
- **Personal user certificate (personal or client certificate)** – an electronic certificate issued to an individual or end user for the purpose of verifying his/her identity and e-mail security (digitally signing e-mails and encrypting their content). It contains data that identifies the individual, such as the username within the identity federation, the full name of the user, e-mail address, etc.
- **Server certificate – an electronic certificate issued to a legal entity** or an institution for the purpose of verifying the identity of a server or service in its ownership. It contains data that identifies the service, most often the Fully Qualified Domain Name of the server.
- **Certificate Service Request (CSR)** – a standard format (according to the recommendation of PKCS #10) used for sending certificate service requests.
- **Certification** – the process of issuing a digital certificate.
- **Certificate serial number** – a number assigned to a certificate, which is unique for each certificate issued by a specific CA.
- **Certificate Revocation List (CRL)** – a list issued and digitally signed by a specific CA that contains the serial numbers of revoked certificates and the time of their revocation. This list needs to be taken into account in the process of verifying the validity of a certificate.
- **Certificate revocation** – the permanent revocation of the validity of a specific certificate and its inclusion in the CRL list.
- **Repository** – a database and/or folder that contains basic documents concerning the work of a specific CA, including any other information pertaining to the provision of certification services by the CA (such as the publication of all issued certificates).
- **Public-Key Cryptography Standards (PKCS)** – a group of public-key cryptography standards defined by the RSA Laboratories.

3 Cryptographic Protocols and Techniques

This section addresses how to ensure secure communication in computer networks, as well as the confidentiality of communication, the preservation of integrity and the authentication of participants in the communication.

3.1 Ensuring Confidentiality – Cryptographic Systems

Only the participants in the communication (the sender and the recipient) should be able to understand a communication whose confidentiality or integrity is preserved. The confidentiality of a communication is achieved by encrypting the messages.

Figure 3.1 shows a general block scheme of the encryption system. The encryption system contains an encryption block, a decryption block, a key generator and the set of messages they emit. The encryption block uses a mathematical function (encryption algorithm) that transforms a set of communicated messages into an unrecognisable form. The encryption algorithm is applied to the original message in combination with the key that represents a sequence of symbols aimed at additionally contributing to the change of the original message. The key is independent of the original message and the function itself and it is generated in the key generator. The encryption algorithm can be considered secure if the security of the encrypted message depends solely on the confidentiality of the key, rather than the confidentiality of the algorithm. A message modified (encrypted) in this manner is then sent through a non-secure telecommunication channel towards its destination. The message can still be intercepted by an interceptor in order to gain an advantage, either through the information contained in the message or by changing the original message and planting false information.

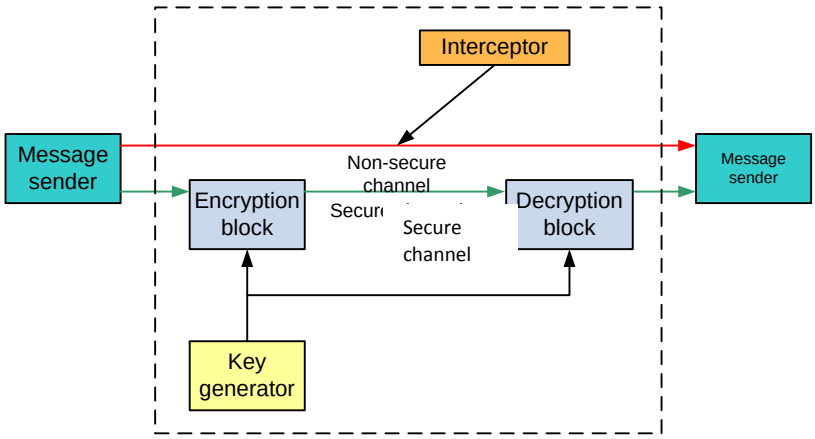

Figure 3.1: A general block scheme of the encryption system

3.1.1 Symmetric-Key Encryption Systems

Symmetric-key encryption systems (symmetric-key cryptography algorithms) are systems that use identical keys for both encryption and decryption so that it is necessary for the sender and the recipient to agree in advance on the keys that will be used for encrypting/decrypting the data. The problem with these systems arises in the distribution and scalability of the symmetric (shared) key.

Symmetric cryptography algorithms ensure the confidentiality of communication, though they do not enable the verification of message integrity or the authentication of its sender.

3.1.2 Asymmetric-Key Encryption Systems

Attempts to resolve the problem of the scalability and distribution of keys between participants in communication have resulted in the development of asymmetric-key systems (asymmetric cryptography algorithms). In these systems, each party has a pair of keys - a public and a private key, where the data encrypted by the public key is decrypted by the private key and vice-versa. The private key is secret and available to its owner alone, while the public key is available to everyone. These keys are generated in such a way that, although they are mathematically connected, it is uneconomic in computer terms to try and detect the private key based on the known public key (in a reasonable period of time). The private key is never communicated over a non-secure channel.

The use of asymmetric-key systems not only ensures the confidentiality of communication, but also the authentication of the sender.

The confidentiality of data is ensured in such a way that the sender performs encryption using the private key of the recipient. Thus, only the intended recipient can decrypt the message since he/she is the only one that has the corresponding private key.

In order to ensure the authentication, the sender encrypts the data sent by way of the private key. When the recipient receives the message, it is decrypted using the public key of the sender. In this way, the recipient is certain that the message was sent by the person who owns the corresponding private key. The procedure represents the basis of the digital signature.

3.1.3 Combined Encryption Systems

The proper combination of symmetric-key and asymmetric-key encryption systems results in combined encryption systems that incorporate their best features. Symmetric cryptography algorithms are simpler and thus enable work at higher speeds by using smaller size keys, while the scalability and distribution of keys are achieved using asymmetric cryptography algorithms, which are more complex and require more processing time.

Figure 3.2 shows a block scheme of a combined encryption system.

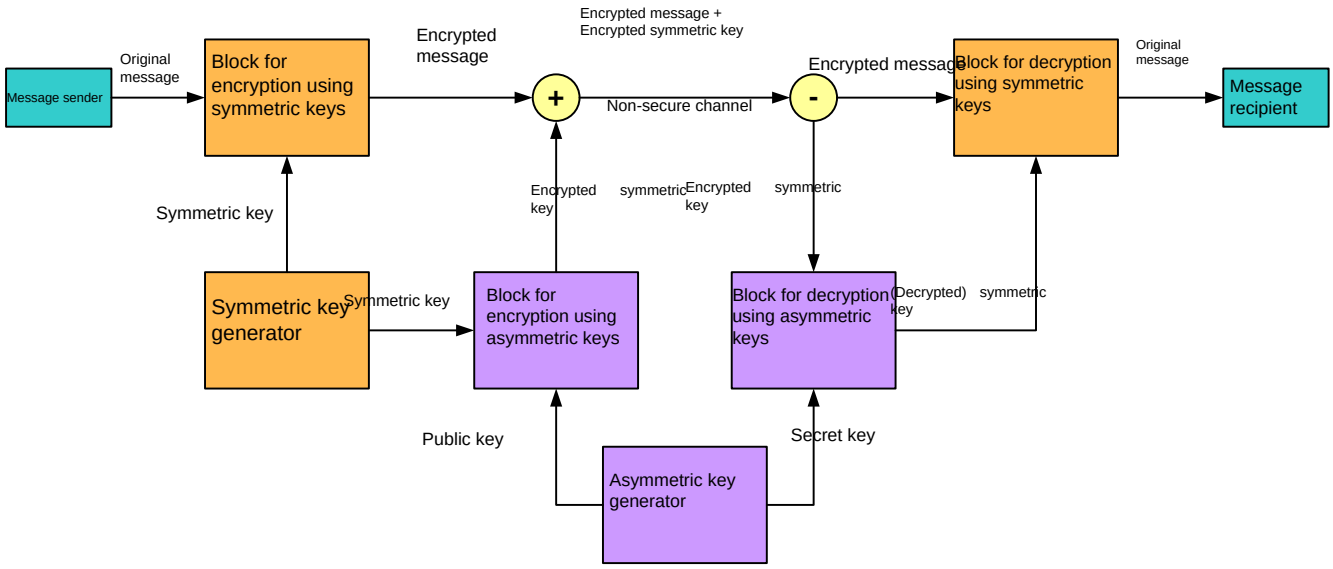

Figure 3.2: A block scheme of a combined encryption system

3.2 Integrity Verification – Hash Functions

The recipient needs to be certain that the content of the message received is the same as the content of the sent message. This is established by verifying the message integrity. A one-way hash function is a cryptographic technique used for verifying the integrity of a message and forms an integral part of numerous security protocols.

The result of applying the cryptographic hash function to a variable-length data block is a sequence of fixed-length bits, also known as the hash value of the message, the message digest or the digest. The basic characteristic of the hash function is that even the smallest change in the original message results in a change to its hash value.

An ideal cryptographic hash function should have the following properties:

- calculation of the hash value of a message is easy;
- it is impossible (in a finite number of steps) to find the message with a given hash value. Algorithms that have this characteristic are called one-way algorithms;
- if an m message is known, it is impossible (in a finite number of steps) to find another m message with a matching hash value (weak collision resistance);
- it is impossible (in a finite number of steps) to find two different messages that have the same hash value (strong collision resistance).

Cryptographic hash functions are used in order to verify the integrity of data transferred over a non-secure channel.

Table 3.1 contains a list of the hash functions that are in use and, along with the algorithm name, the length of their hash values. The longer the hash, the more the algorithm is resistant to various types of attack and the better security it provides.

Algorithm name	Hash length
GOST	256 bits
HAS-160	160 bits
HAVAL	128-256 bits
MD2	128 bits
MD4	128 bits
MD5	128 bits
RadioGatun	Up to 1216 bits
RIPEMD-64	64 bits
RIPEMD-160	160 bits
RIPEMD-320	320 bits
SHA-1	160 bits
SHA-224	224 bits
SHA-256	256 bits
SHA-384	384 bits
SHA-512	512 bits
Skein	256, 512, 1024 bits
Snefru	128, 256 bits
Tiger	192 bits
Whirlpool	512 bits

FSB	160-512 bits
ECOH	224-512 bits
SWIFFT	512 bits

Table 3.1: Hash functions

3.3 Verifying Integrity with Authentication – MAC Code

A Message Authentication Code or MAC basically has the characteristics of a hash with the addition of a shared key, which, besides verifying the message integrity, also enables the authentication of the sender. Like the hash function, a MAC is a sequence of fixed-length bits achieved by applying the MAC algorithm to the message and the secret key known to both parties in the communication. The use of the key ensures that only the person with an identical key can verify the hash function. Thus the owner of a message/file can verify its authenticity if he wants to be sure that the file has not been changed due to an attack (virus activity, etc.), while the message recipient can identify its sender.

There are two categories of MAC code depending on the type of algorithm used:

- **HMAC** or Hash-based Message Authentication Code – achieved by applying a known hash algorithm in the implementation of the MAC algorithm (HMAC-MD5, HMAC-SHA1),
- **CMAC** or Cipher-based Message Authentication Code – achieved by applying symmetric block cipher in a cipher block chaining mode.

The hash and MAC values of the message enable the verification of its integrity, while the MAC also ensures the authentication of the message sender. However, neither MAC nor hash provide non-repudiation protection, i.e. they do not ensure that the message sender cannot subsequently claim he/she did not send the message or that someone else sent the message on his/her behalf. Digital certificates are used to ensure non-repudiation protection.

3.4 Verifying Integrity with Authentication and Non-Repudiation – Digital Signature

Authentication with non-repudiation ensures that the participants in a communication are actually the persons they claim to be. Asymmetric-key encryption algorithms can ensure the verification of integrity and identity (authentication) using digital signatures.

The digital signature of electronic data can be viewed as analogous to the signature or stamp on printed documents. It allows the message recipient to be certain that the original content of the message has not been changed and to be certain about the identity of the message sender. In other words, it guarantees the integrity of the message and the identity/authentication of its sender. Also,

the digital signature cannot be denied so the person who sent the message cannot subsequently claim that he/she did not send it or that it was sent by some other person.

The digital signature of the message is created using the asymmetric key technique. The sender creates a hash (an encrypted summary of the message) by applying the hash algorithm to the original message. The hash of the message is a “digital fingerprint” of the message. If the original message is even slightly changed, the resulting hash of the message would also be changed. The sender then encrypts the hash with his/her own private key. The hash encrypted in this way represents the digital signature of the message.

The sender adds the digital signature at the end of the original message and sends the message digitally signed in this manner. Upon receipt, the recipient decrypts the digital signature of the message using the public key of the sender in order to get the hash of the sent message and then compares it with the value of the hash received by applying the same hash algorithm to the received message itself. If the hash values obtained are identical, the recipient can be certain the message has not been changed. If the hash values are different, it means that there has been an unauthorised change to the content of the message and/or the person who sent the message is not the person he/she claims to be.

Figure 3.3 shows a block scheme of a system with a digital signature.

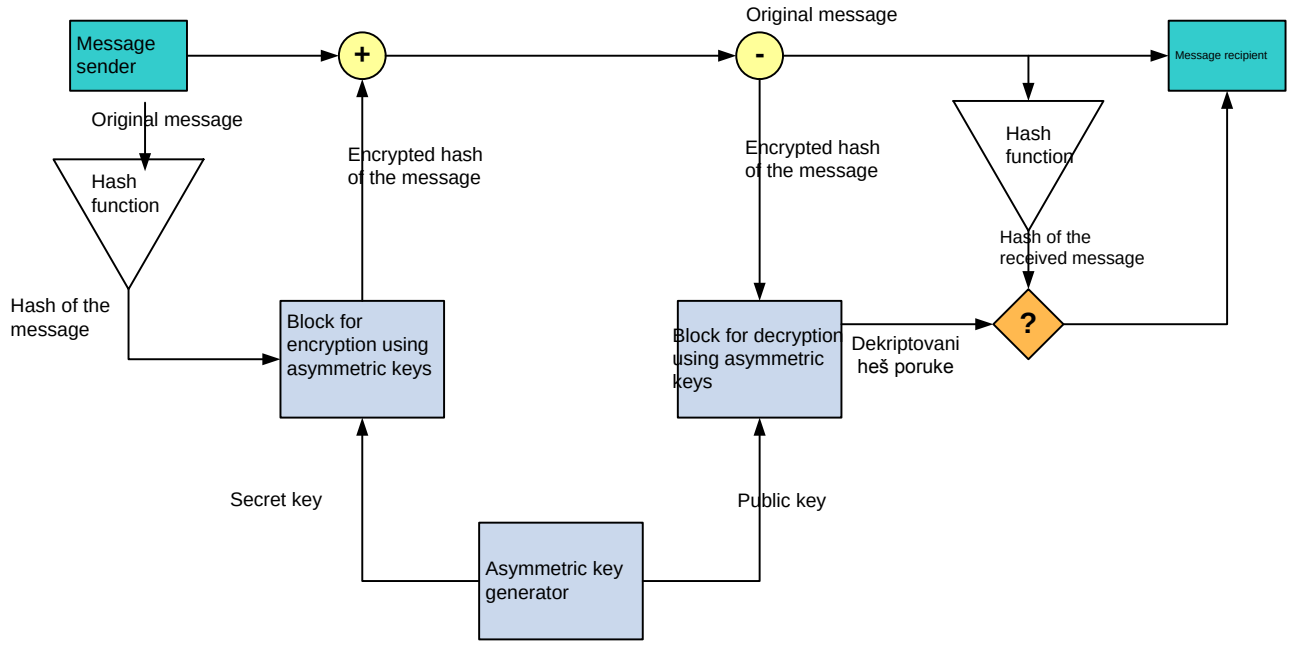

Figure 3.3: A block scheme of a system with a digital signature

3.5 Digital Certificates and the Public Key Infrastructure

When using the asymmetric cryptography technique, either to ensure the confidentiality of information by encryption, as a digital signature for the purpose of authenticating the sender or to ensure the integrity of data, there is a problem concerning the authentication of the asymmetric key pair itself. The basic question that arises here is how it can be guaranteed that the public key does indeed belong to the person we believe is its owner. A potential attacker can plant his/her own public key in order to read a message intended for someone else or to compromise the integrity of the message by signing it with another private key. This problem is solved by using digital certificates and the Public Key Infrastructure.

A digital certificate (Public Key Certificate – PKC) enables the identity verification of the participants in electronic communication in a way that is similar to identity cards in human interactions. It links the data on the identity of the participants in communication with the pair of asymmetric keys used to encrypt and sign digital information, which enables the confirmation of a person's right to use the cryptographic key pair. Specifically, it enables confirmation that a certain public key belongs to a certain end entity (end-user or end-server). This prevents the possibility of the keys being abused or that an unauthorised person assumes someone else's identity.

The information contained in the digital certificate is digitally signed and thus confirmed by the Certification Authority (CA). The Certification Authority is responsible for issuing certificates and it is an indisputable authority trusted by all the participants in the communication. The concept that integrates the use of the digital certificates and the role of the certification authorities is called the Public Key Infrastructure and is explained in the next chapter.

A digital certificate contains data on the owner / end entity of the certificate, the public key of the owner / end entity of the certificate, the period of validity of the certificate, the name of the issuer (the Certification Authority that issued the certificate), the serial number of the certificate and the digital signature of the issuer.

The most commonly used format of digital certificate is defined by the ITU-T X.509 standard, version 3. It can be stored separately from the private key (PEM format), or it can be stored together with the private key in various formats (e.g. PKCS #12, JKS, etc.).

When sending a message, the sender digitally signs the message and sends the digital certificate along with the message so that the recipient can be certain of the identity of the sender. The message can also be accompanied by several digital certificates that form a certificate chain, or the hierarchy chain of trust, where one certificate confirms the authenticity of the previous digital certificate. At the very top of the hierarchy of trust is the root (top-level) Certification Authority, the so-called Root Certification Authority, trusted by all other certification authorities. The public key of the Root Certification Authority needs to be publicly known and available.

Figure 3.4 shows a block scheme of a system with digital certificates.

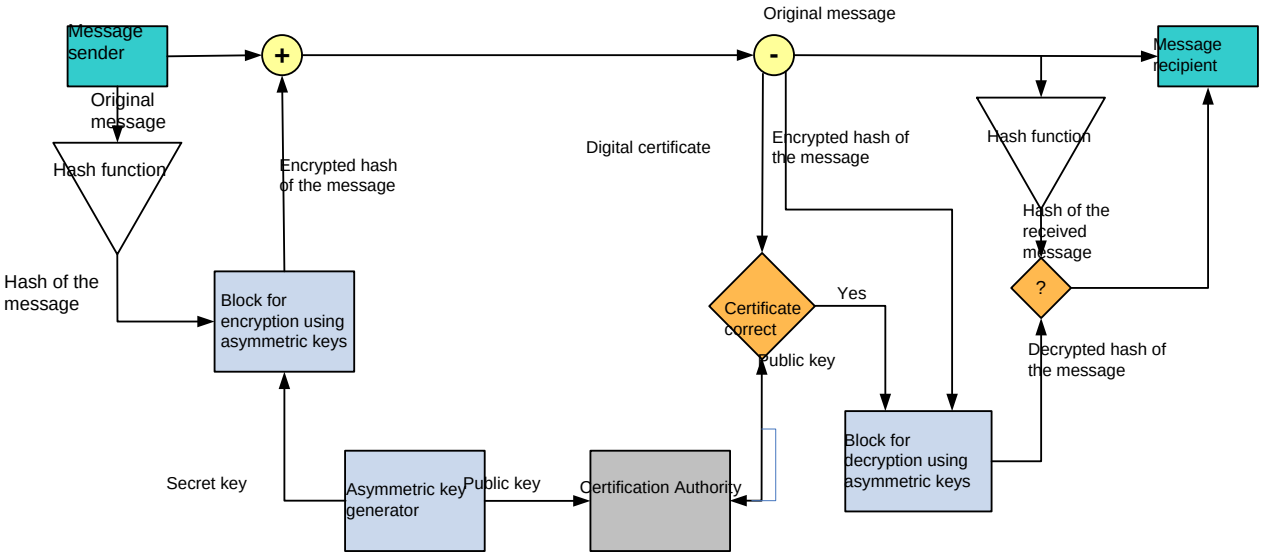

Figure 3.4: A block scheme of a system with digital certificates

4 Public Key Infrastructure (PKI)

The Public Key Infrastructure (PKI) contains hardware, software, policies and procedures needed to manage, generate, store and distribute cryptographic keys and digital certificates. It defines the model for a comprehensive security infrastructure whose services are implemented by way of the encryption system concept that uses asymmetric algorithms.

The Public Key Infrastructure is defined by the ITU-T X.509 standard, accompanied by the IETF document RFC 5280, which provides more detailed recommendations for the Internet community.

4.1 PKI – Its Components and Basic Functions

The Public Key Infrastructure (PKI) comprises the hardware, software, policies and procedures needed to manage, generate, store, distribute, use and revoke cryptographic keys and digital certificates.

Figure 4.1 shows the interrelation between the basic PKI elements.

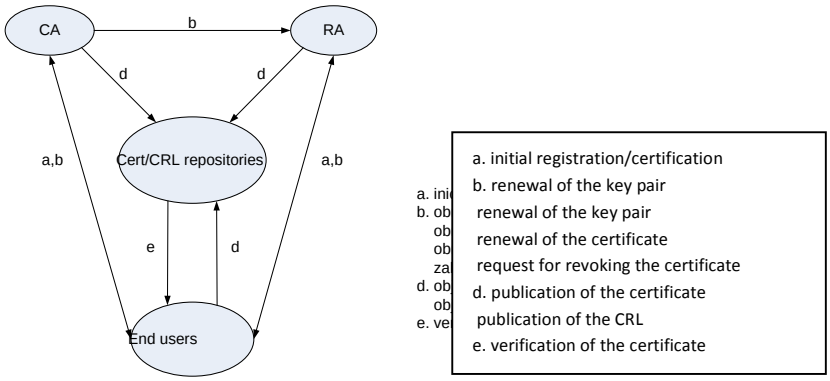

Figure 4.1: Relations between PKI elements

Cryptographic keys and digital certificates are managed, generated, stored and distributed through **Certification** (**CA**) and **Registration** (**RA**) **Authorities**.

The participants in a communication who do not have an established chain of trust rely on a trusted third party - the Certification Authority (CA). The CA, as a trusted authority, has the capacity to issue and revoke digital certificates (Public Key Certificates – PKC). Before issuing a certificate, the CA undertakes a complete check of the data on the owner / end entity for whom the request for issuing the certificate has been submitted. The CA can perform these checks directly or via one or more RAs. Following the successful check of data, the CA issues a digital certificate to the owner / end entity. The issued certificate guarantees that the public key belongs to that specific owner/end entity. The end entity can be an end user (individual) or a server, while the digital certificate can link the public key to the identity or the DNS record of the end entity. A digital certificate issued to an end user guarantees its identity and is signed by the digital certificate of the Certification Authority itself. End users now use their own digital certificates, issued by a trusted Certification Authority, in order to prove their identity to one another and to establish secure communication.

Digital certificates are issued to both individuals and legal persons. Individuals (end users) are issued with personal certificates. Legal persons, i.e. institutions, are issued certificates for the end entities they own, most often server SSL certificates for user servers (web, e-mail, RADIUS, etc.).

4.1.1 Registration, the Application Process for Institutions/Users

In order to use the services of the Public Key Infrastructure, institutions / end users first need to go through an application process that includes the verification of their identity and the exchange of information with the appropriate component of the infrastructure – the **Registration Authority** (**RA**). The first step in the application process is registration, which involves the verification of the identity of an end entity that has submitted a certificate request. With respect to the end user, the registration means the verification of the identity of the end user who has submitted a certificate request. In the case of institutions, the request is issued for its user servers. For that reason, the procedure for institutions also includes a pre-registration stage that involves the verification of the data on an institution and the submission of the names of the persons (as part of the procedure for the designation of the institution's representatives) that will be authorised to submit certificate requests required by the institution. The level of the verification of identity of an end user or an institution depends on the type of certificate requested (the certificates used to ensure the security of electronic financial transactions require a stricter level of verification than other types of certificates). The appropriate level of verification is defined for each type of certificate by a document called the **certificate policy**.

4.1.2 Initialisation

The next step in the registration process is initialisation, during which the representative of an institution/end user and the Certification Authority exchange the information necessary for further communication, such as the manner in which the communication will be performed, the manner of distribution of the certificates, the manner of establishing secure communication between the representatives of the institution / end users and the Certification Authority, the manner of generating and delivering an asymmetric key pair, etc.

4.1.3 Certification

The certification process involves issuing and delivering certificates to the representatives of an institution/end user and is conducted by the CA.

4.1.4 The Revocation of a Certificate

While the validity period of a digital certificate is defined by the dates inserted in the appropriate fields of the certificate, it can happen that the secret key is compromised or some of the basic personal data on the certificate is changed, which requires the revocation of the certificate, i.e. its withdrawal from use. The revoked certificate is included in the **Certificate Revocation Lists** (**CRLs**) published by the Certification Authority that issued the certificate. By using the **Repository** – a database and/or folder that contains basic documents on the work of the specific CA – the CA publishes any other information pertaining to the provision of certification services by the CA (such as the publication of all issued certificates, etc.).

In some cases, real-time certificate status protocols are used instead of the Certificate Revocation Lists, which provide a positive or negative response concerning the status of a certificate. An example of such a protocol is the Online Certificate Status Protocol (OCSP).

However, the recipient of a certificate also has the duty to verify the certificate before accepting it. The verification of a certificate is not an easy process taking into account that the signatory of a message can provide a chain of certificates instead of one, i.e. its own certificate, where each certificate is signed by the certificate of the superior CA. This implies the verification of the chain of trust and the validity of each certificate contained in the chain.

4.1.5 Verifying the Chain of Trust

Certificate verification of each individual certificate must provide answers to the following questions: Is the given certificate trusted? Is the certificate actually signed by the specific CA?

If the recipient of the certificate does not trust the first certificate in the chain, the recipient needs to verify the next certificate in the chain (i.e. the certificate of the CA that has signed it). The process continues until the recipient finds a trusted certificate in the chain, i.e. a trusted CA that owns the certificate.

4.1.6 Verifying the Validity of a Certificate

The verification of the validity of each individual certificate needs to provide answers on whether the specific certificate has expired and whether the certificate is still valid or has been revoked. As mentioned above, invalid (revoked) certificates are published on Certificate Revocation Lists or certificate status protocols are used to verify the status of a certificate. Each certificate contains a field

that specifies its validity period and shows whether the certificate has expired. Certificates are usually issued for a period of one to two years.

4.2 The Format of Digital Certificates

The format of a digital certificate is defined by the ITU-T X.509 standard. Today, the most commonly used version is Version 3.

According to the recommendations of IETF RFC 5280 (Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile), a digital certificate should contain the following fields (see Figure 4.2):

- **Version** – information on the version of the certificate. The current version (and the one that is most commonly used) is Version 3.
- **Serial number** – the certificate serial number is the unique identifier of certificates issued by a specific Certification Authority.
- **Signature** – an identifier in the OID (object identifier) format. It identifies the algorithm used by the Certification Authority for signing the certificate.
- **Issuer** – a unique name of the Certification Authority that issued the certificate. This is provided in the Distinguished Name (DN) format.
- **Validity** – specifies the period of validity of the certificate and contains two dates, one of which indicates the time from which the certificate becomes valid (the field *Not Valid Before)*, while the other indicates the time until which the certificate can be considered valid (the field *Not Valid After)*.
- **Subject** – identifies the user/entity the certificate has been issued to. As is the case with the *Issuer* field, it is provided in the format of the Distinguished Name (DN) of the entity, e.g. "C=RS, O=University of Belgrade OU=RCUB, CN=Milica Kovinic".
- **Subject public key info** – contains the public key of the user/entity and the identification of the algorithm used in creating the public key (RSA, DSA, Diffie-Hellman, etc.).
- **Unique identifiers** – an additional identifier of the certificate whose usage is not recommended in versions 2 and 3.
- **Extensions** – contains a list of one or more certificate extensions used for specifying additional information about the certificate. Each item on the list contains information on the type of extension, the critical/non-critical mark and the extension content. If the user software cannot recognise the extension during processing, the certificate will be marked as invalid if the extension is marked as a critical one. If the extension is marked as non-critical, it can be ignored and the offered certificate can be accepted. The extensions in use belong in the following categories:
    - Standard extensions:
        - **Authority key identifier** (**AKI**) – enables identification of the public key that corresponds to the private key of the Certification Authority, by which the

certificate has been signed. It is usually used when the Certification Authority has more than one key for signing.

- **Subject key identifier** (**SKI**) – a unique identifier used for identifying certificates that contain the corresponding public key.
- **Key usage** – defines the purpose of the key contained in the specific certificate, i.e. whether it can be used for signing, encryption, verification of the Certificate Revocation Lists, etc.
- **Certificate policies** – contains a list of identifiers of certificate policies (provided in the OID – object identifier format) and additional optional textual parameters confirming that the specific certificate has been issued under an appropriate set of rules incorporated in the specified certificate policies. Optional parameters include the Certification Practice Statement pointer (which defines the location where the Practice Statement of the Certification Authority can be found) and the User notice (which contains the text of the message for the end user).
- **Policy mappings** – used in certificates of the certification authorities in various administrative domains in order to indicate the set of pairs of certificate policies that are considered equivalent. A pair of certificate policies to be specified in this extension is indicated in the fields *Issuer domain policy* and *Subject domain policy*.
- **Subject alternative name** – contains additional data on the certificate subject/object (e.g. the end user's e-mail, URL, IP address, alternative DNS name of the server, etc.).
- **Issuer alternative name** – contains data on the Certification Authority that issued the certificate (e.g. e-mail, URI, IP address, etc.).
- **Subject directory attributes** – provides additional information on the certificate subject (e.g. nationality) and is not of great importance.
- **Basic constraints** – provides information on whether the certificate subject is a Certification Authority or an end user/entity. In the case of a Certification Authority, the extension can also contain the *Path length* parameter (showing the number of certificates of certification authorities that can follow it).
- **Name constraints** – used only in certificates for certification authorities; defines the space of the name provided in the form of *Permitted subtrees* and/or *Excluded subtrees*, to which the types of names in the certificates of end users issued by those certification authorities must/must not belong.
- **Policy constraints** – used in certificates for certification authorities in order to prevent mapping of certificate policies or to indicate the need to include a certain certificate policy in all the certificates that are verified after the certificate with this extension. The presence of the *inhibitPolicyMapping* field suggests that the mapping of certificate policies is forbidden after a certain number (defined by the value of this field), while the presence of the *Require explicit policy* field suggests the usage of a certain certificate policy in a certain number (defined by this value) of certificates that follow.

- **Extended key usage** – used in certificates for end users/entities to indicate additional possibilities for key usage (besides those defined by the *Key usage* extension).
- **CRL distribution points** – indicates the locations of the Certificate Revocation List.
- **Inhibit anyPolicy** – used in certificates for certification authorities to define the relationship towards a special value of a certificate policy – *anyPolicy* (which specifies the possibility of using any certificate policy) with the next certification authorities. The value refers to a number of additional certificates that can appear in the chain before *anyPolicy* is no longer allowed, i.e. the number of certification authorities after this, whereupon *anyPolicy* is no longer considered valid in terms of matching it with another specific policy.
- **Freshest CRL pointer** – points to the "freshest" information concerning revoked certificates. In practice, this is usually the pointer towards the Delta list of revoked certificates.

o Private extensions are defined during the usage of certificates in specific fields. For instance, RFC 5280 specifies two extensions of this type that can be applied for the Internet:

- **Authority information access** – defines the manner of access to services (and information) provided by the Certification Authority issuing a certificate (e.g. about the service for online verification of the certificate status, additional information about the certificate policy, etc.).
- **Subject information access** – defines the manner of access to services (and information) provided by the certificate subject (e.g. the location where the certificate is stored when the certificate subject is a Certification Authority).

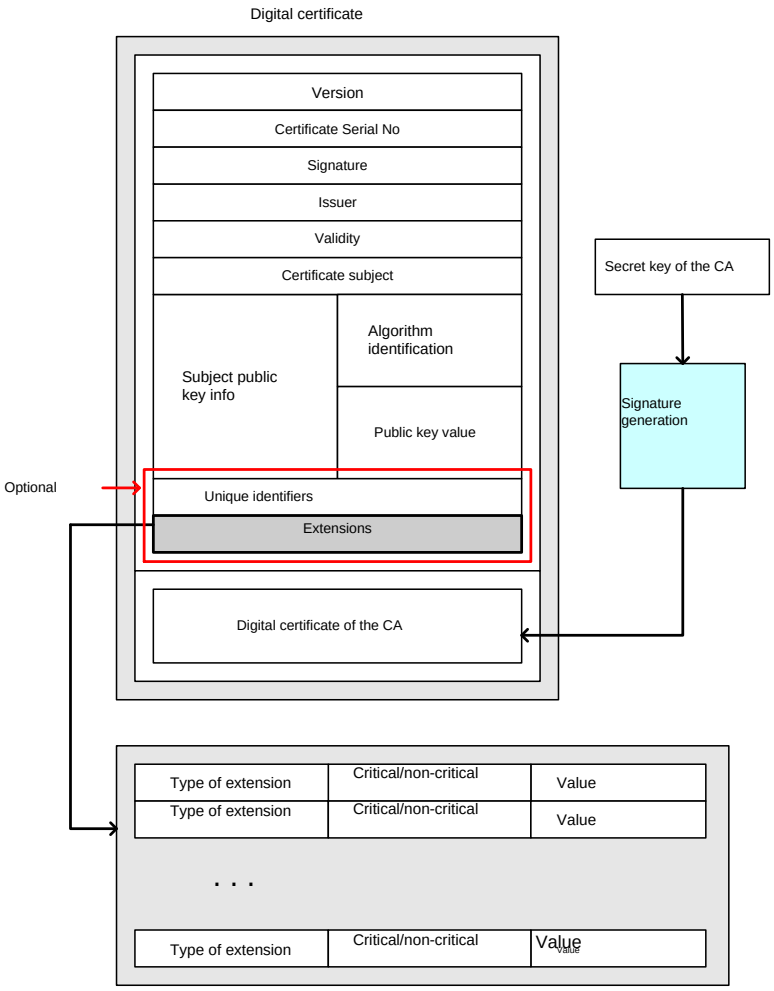

Figure 4.2: Digital certificate according to RFC 5280

A digital certificate can be stored in the PEM format, separately from the private key, or it can be linked to the private key in various formats (e.g. PKCS #12, JKS, etc.).

The most commonly used file extensions for X.509 certificates include:

- .CER, .CRT, .DER: DER (Distinguished Encoding Rules) format, most often in a binary form, though it can also be Base64.
- .PEM: Base64 encoded DER certificate, located between "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----".
- .P7B: see P7C.
- .P7C: PKCS #7 format, contains only certificates (one or more), though it can also be used for the CRL.
- .PFX: see P12.

- .P12: PKCS #12 format, contains a certificate and the corresponding private key (protected by a code).
- .CSR:PKCS #10 format, contains the certificate request (CSR).

5 The TCS – TERENA[1] Certificate Service

The TCS (*TERENA Certificate Service*) issues digital certificates to scientific, research and education institutions through their national academic networks. TERENA has the role of a Certification Authority, while the Registration Authority role is entrusted to the NREN members (of the Academic Network) registered to use the TCS service. The certificates obtained through the TCS service are issued by *Comodo CA Limited*[2], one of the leading and globally recognised certification authorities.

5.1 Types of Certificates Offered by TCS

TCS offers five different types of digital certificates:

- **Server SSL Certificates (Server Certificates)** – also called SSL certificates for authenticating servers and establishing secure sessions with end clients. The validity period of these certificates is up to three years. There are three types of Server Certificates, depending on the registered DNS name of the server included in the certificate:
  - **Single SSL Certificates** – this type of certificates is linked to only one registered DNS server name, which is included in the certificate as the value of the CN (Common Name) attribute.
  - **Multi-Domain SSL Certificates** – this type of certificate ensures more than one registered DNS name by defining one name as primary and placing it in the CN field of the certificate, while each additional name is defined as a value of the SAN (Subject Alternative Name) field. This certificate can contain up to 100 different DNS names of services located in one physical machine (server).
  - **Wildcard SSL Certificates** – one certificate allows an unlimited number of subdomains located on different physical machines (servers). This certificate contains the domain name in the form *.some_domain (e.g. *.rcub.bg.ac.rs) in the CN field and thus protects all the subdomains under the defined domain. Taking into account that the security of the entire domain is covered by one certificate, the usage of a Wildcard Certificate is only justified in the following cases:

[1] From October 2014, TERENA and DANTE were combined into the GÉANT Association.
[2] The agreement between TERENA and Comodo CA Limited is valid until July 2015, after which the CA role in the TCS service may be taken over by another certification authority.

- web server farms,
- clusters,
- load balancers,
- failover servers.

- **e-Science Server Certificates** – for authenticating GRID hosts and services. The goal is that they are IGTF compliant. The validity period of these certificates is up to 13 months.
- **Personal Certificates** – also called Client Certificates, are used for authenticating or identifying end users accessing user servers/services and securing e-mail communications (digital signing of e-mails and encryption of their content). These certificates contain data that identifies the end user, such as the user name within the identity federation, the user's full name, e-mail address, etc. The validity period of these certificates is up to three years.
- **e-Science Personal Certificates** – for authenticating or identifying end users accessing Grid services. They are intended to be IGTF compliant. The validity period of these certificates is up to 13 months.
- **Code-signing Certificates** – for authenticating software distributed over the Internet. The validity period of these certificates is up to five years.

As of 1 February 2013, the TCS service offers two types of server certificates (Server SSL and e-Science Server Certificates), depending on the identification data they contain:

- **Organisation Validated (OV) Certificates** - in addition to the DNS server name, the OV Certificates contain the common name (CN) and information on the organisation for which the certificate is issued. More specifically, the additional information this certificate contains is the name of the country (C), the name of the organisation (O), and the name of the organisational unit within which the certificate will be used (OU). Bearing in mind the larger amount and type of identification data in the certificate, the procedure for their verification before issuance is significantly more complex and takes more time.
- **Domain Validated (DV) Certificates** - this type of certificate contains only the DNS name (or names) of the server. They do not include information about the organisation they are issued to. Therefore, the only identification data that is validated before their issuance is the domain to which the name of the server belongs. The domain verification procedure is simple and quick, which comprises the main advantage of this type of certificates.

5.2 SHA-2 Certificates

Until recently, only the SHA-1 algorithm has been used for signing certificates. However, this algorithm is not entirely secure. The first signs of weakness in the SHA-1 were identified as early as 2005.[3] Later, in 2012, the research showed that it was possible to compromise this algorithm, though it was still

[3] https://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html

costly to do so.[4] In November 2013, Microsoft Corporation announced that from 2016 it would no longer accept certificates signed using this algorithm.

In line with these developments, the TCS service has, since October 2014, enabled the issuance of certificates that rely on the SHA-256 algorithm for signature (SHA-2 certificates). SHA-2 certificates can now be requested for all the types of certificates described in the previous chapter.

All requested certificates whose validity period expires after 1 January 2017 will be signed using the SHA-256 algorithm, regardless of whether such a signature has been specified in the Certificate Signing Request (CSR). For the issuance of certificates signed with the SHA-256 algorithm whose validity period expires before 1 January 2017, it is necessary to specify the desired hash algorithm in the Certificate Signing Request.

5.3 The Advantages of TCS Certificates

The certificates obtained through the TCS service belong to the chain of trust whose root certificate is preinstalled in most of SSL clients (e.g. when accessing a website via https, which is located on a web server that has a TERENA certificate, no intervention of the user is required in order for the certificate to be accepted since the corresponding root certificate is already preinstalled in most operating systems). Currently, the SHA-1 certificates are signed by the TERENA CA certificate, which is further signed by the certificate of UserTrust, an intermediate Certification Authority signed by the AddTrust External Root Certification Authority. The SHA-2 certificates, which can now be requested, belong to a somewhat different chain of trust. The SHA-2 certificates are signed by the TERENA SSL CA 2 certificate, which is further signed by the certificate of the UserTrust RSA Certification Authority, while the root certificate is still signed by the AddTrust External CA Root certificate.

The AMRES services of issuing TCS certificates (the AMRES TCS service) comprise the following:

- **Server Certificate Issuance Service**, through which the AMRES user institutions can obtain all types of server certificates; and
- **Personal Certificate Issuance Service**, through which the AMRES user institutions can obtain personal user certificates.

It should be noted that the certificate policy requires clear and efficient procedures for data verification to be established in the course of issuing and using certificates. These procedures are established for both server certificates and personal user certificates.

Server SSL certificates can be used on all AMRES servers that are not part of the GRID infrastructure. The issuance of e-science of certificates (server and personal) for protection and access to GRID installations is currently enabled through the AEGIS CA. The AEGIS CA has been formed in order to provide PKI services for the GRID research community in Serbia. The AEGIS policy is published in a

[4] https://www.schneier.com/blog/archives/2012/10/when_will_we_se.html

document that can be found on: http://aegis-ca.rcub.bg.ac.rs/documents/AEGIS-CP-CPSv1-2.doc, which also contains the Practical Statement of the AEGIS CA.

5.4 Services that Need to Be Secured with Digital Certificates

Digital certificates should be used for:

- **The authentication of servers** (web, e-mail, RADIUS, etc.) by a client or another server. In this case, server SSL certificates are used to confirm the identity of a server to the client accessing it. The client can thus be certain it has accessed the right server, which is important if sensitive data such as user credentials are being sent. Likewise, in addition to the authentication of servers, these certificates ensure the confidentiality of communication in such a way that the client sends the key that will be used for the encryption of data, which is encrypted by the public key of the server.
- **The authentication of end users / clients** by the server they are accessing. In this case, the client has its personal certificate, which is verified by the server in order to confirm the client's identity. An example of this is the authentication of clients in the GRID infrastructure, where each user has its Personal Certificate used for its authentication for accessing the GRID services.
- **The secure exchange of e-mails** that ensures the preservation of the integrity and confidentiality of communication. The sender digitally signs an e-mail message with his/her own private key, which guarantees the integrity of the message. In order to ensure confidentiality, it is necessary for the participants in the communication to exchange their personal certificates in advance.
- **Establishing an IPsec/TLS VPN tunnel** and the authentication of end users that have initiated the establishment of the VPN tunnel. In addition, in the case of SSL VPNs, the certificates are also used for the authentication of the SSL VPN server by VPN clients. Therefore, in this case an SSL certificate and personal certificates are required.
- **The digital singing of software** ensures proof of the origin and integrity of the software code. When the server is digitally signed with a certificate issued by a trusted Certification Authority, the user can be certain that it has not been maliciously modified and that it is safe to install. This requires a code signing certificate.

Below are the recommendations concerning the commonly used server/services that need to be secured with server SSL certificates. Likewise, the protection of e-mail using personal certificates is explained.

5.4.1 Web Server

The HTTPS Protocol (*Hypertext Transfer Protocol Secure*) is used to ensure secure communication between a web server and a client through an unprotected network, such as the Internet. Secure

communication established in this way enables the client to authenticate the server it accesses, as well as the encryption of data exchanged between the client and the server. HTTPS is an HTTP protocol that uses the services of the TLS/SSL protocol on a lower layer.

For HTTPS communication, the web server needs to be configured on port 443 and use the server SSL certificate. The Server Certificate contains information that enables the client to confirm the identity of the server before sending confidential information (such as usernames and passwords).

The HTTPS connection is established through the following steps:

- The client accesses the web page that requires the entry of some sensitive data, such as user credentials.
- The web server sends its Server Certificate to the client.
- In order to verify the certificate, the web client verifies the local database of certificates on the user's computer in order to find a certificate issued by the Certification Authority of the web server. If the certificate is found, the client can trust the Certification Authority that issued the certificate. Further, the client verifies the name of the server in the certificate, which must correspond to the DNS name through which the client has accessed the server.
- By using the public key contained in the Server Certificate it has received, the web client performs encryption of the data it sends to the server.
- In order that the server can send encrypted messages to the client, the client sends the server its key, which the client generated for that session and which is encrypted using the public key of the server. As of this moment, the secure exchange of data between the client and the server can begin.

If the Server Certificate is not signed by the Certification Authority trusted by the client, a pop-up message appears (Figure 5.1) with a warning for the user that the site being accessed is not secure and the question whether the user wants to continue further communication (which can often confuse the user).

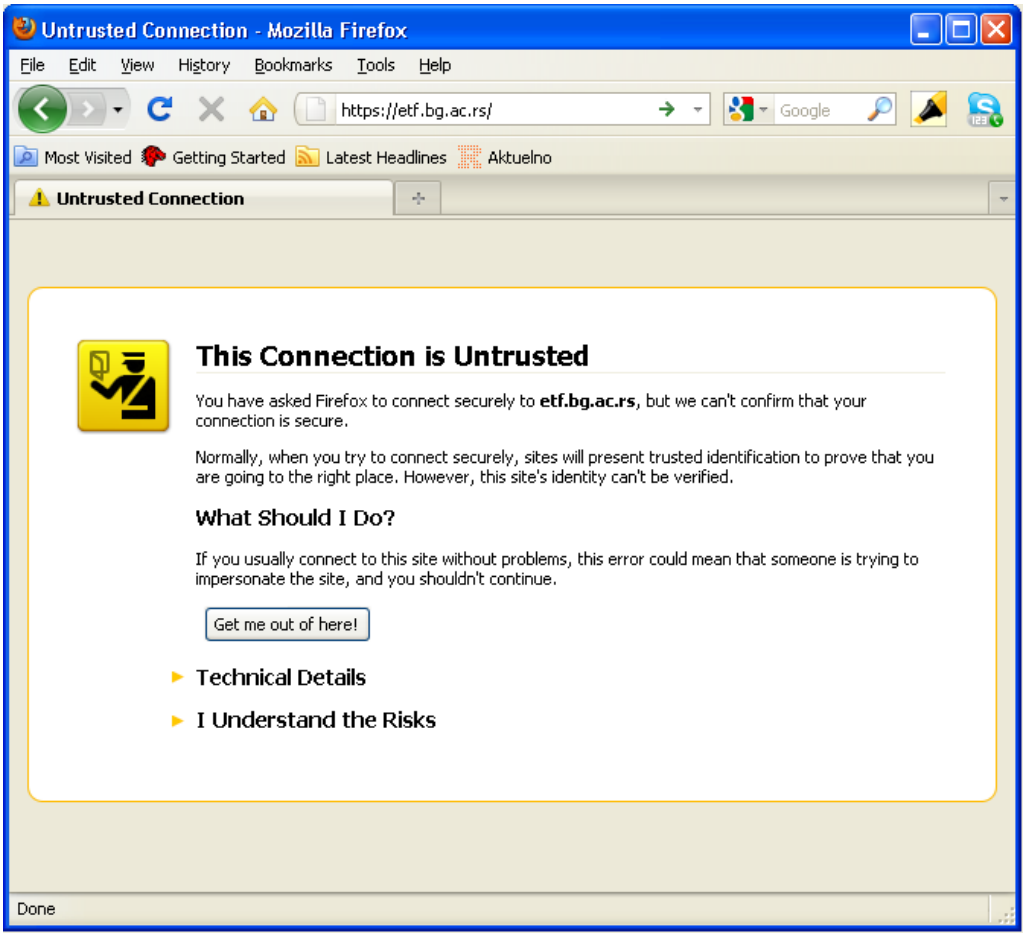

Figure 5.1: A message warning the user that the website being accessed is not safe

In HTTPS communication, clients can also use their personal certificates. The client's certificate contains personal data about the user and enables the server to authenticate the user.

5.4.2 RADIUS Server

Users send their credentials to RADIUS servers in order to be authenticated for accessing network resources. In the AMRES network, the use of certificates on the RADIUS servers within the *eduroam* services is mandatory. The IEEE 802.1x standard is used for the authentication of clients. The EAP Protocol (Extensible Authentication Protocol) is used for conveying authentication messages in combination with TLS, TTLS (*Tunnelled TLS*) or PEAP (*Protected EAP*) protocols.

In the process of authentication, the RADIUS server receives a message from the client that marks the beginning of authentication. The server responds by sending its digital certificate to the client. The client verifies the certificate by attempting to find in its local database the certificate of the Certification Authority that issued the Server Certificate. If the outcome is positive, the client sends its

credentials (identity and password) encrypted by the public keys of the server. The RADIUS server decrypts the received data using the private key and verifies the user credentials in its database.

5.4.3 E-mail Server

STARTTLS is an extension of the SMTP, IMAP and POP3 protocols (SMTPS, IMAPS, POP3S) that enables establishing an encrypted connection with the support of the SSL/TLS Protocol. Since STARTTLS is an upgrade of the existing protocols, no special port is required for encrypted communication. Separate ports are registered for the SMTPS, IMAPS and POP3S protocols, though RFC does not recommend their utilisation since the use of STARTTLS enables the usage of the same port for both protected and unprotected communication.

Besides enabling the confidential, encryption-protected exchange of data, STARTTLS offers the possibility of authentication between servers, as well as between a client and a server. In order to establish encrypted/authenticated communication, the party that initiates the communication sends the *starttls* message in order to mark the transfer to protected data exchange.

Authentication can be two-sided or one-sided. It is often configured so that the client authenticates the server it is accessing (in the case of POP3, IMAP and SMTP protocols), in the same way as when the authentication is performed between servers in the case of the SMTP protocol. The party that performs authentication needs to have its own digital certificate verified by the other party.

In the case of encrypted communication between the client and the server or between servers, the sending party uses the public key from the digital certificate of the other party in order to send an encrypted message. Upon receipt, the other party performs decryption using its private key.

It should be noted here that the protected communication is not performed from one end to the other, but only between servers/clients that are configured to use STARTTLS. One of the ways to ensure the confidentiality of communication between end clients is to use the S/MIME standard, which is explained later in this text.

5.4.4 Electronic Mail Security

MIME is an Internet standard that defines the extended format of an e-mail message. S/MIME (*Secure/ Multipurpose Internet Mail Extensions*) is an extension of the MIME standard, which ensures the protected exchange of e-mails that enables the authentication of the sender, proof of the integrity of the message, and the encryption of its content. In this case, both parties in the communication need to have digital certificates and to exchange them.

Figure 5.2 shows the S/MIME encryption/decryption of a message. In order to encrypt a message, the sender generates a symmetric key for that session, which encrypts the body of the message. The sender then encrypts the symmetric key using the public key of the recipient. Upon receipt, using a private key, the recipient decrypts the symmetric key generated for that session, which is used to

decrypt the body of the message. This results in the confidential exchange of e-mail messages, where the entire process is only fully transparent for the end users.

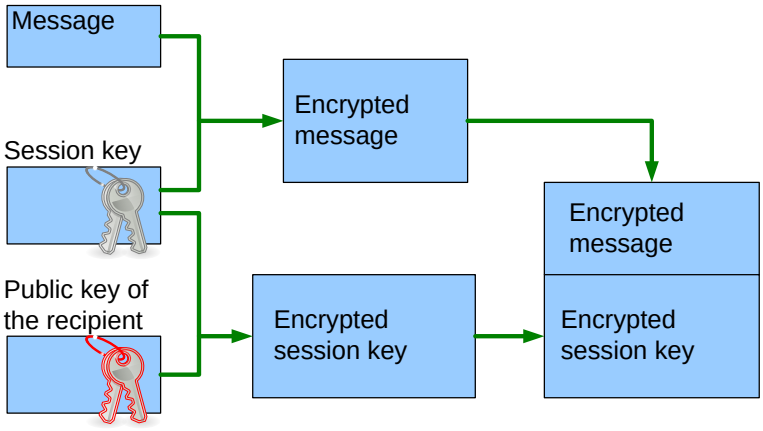

Figure 5.2: S/MIME encryption/decryption of a message

In order to ensure the integrity of a message, a digital signature is created and sent along with the message. The digital signature of the message is obtained by calculating its hash value. Taking into account that the sender's public key is used for decrypting the hash, this confirms that only the person who has the corresponding private key was able to send the message, which completes the authentication of the party that has sent the message.

6 The AMRES TCS Certificate Service

In cooperation with TERENA, AMRES has established a service for issuing digital certificates, where TERENA has the role of the Certification Authority (CA) and AMRES has the role of the Registration Authority (RA). The TCS service is free of charge for all members of AMRES who have completed the registration process.

The AMRES services of issuing TCS certificates (the AMRES TCS service) comprise the following:

- **Server Certificate Issuance Service**, through which the AMRES user institutions can obtain all types of server certificates; and
- **Personal Certificate Issuance Service**, through which the AMRES user institutions can obtain personal user certificates.

The Certificate Policy prescribes the procedures for verifying data in the process of issuing and using certificates. The procedures defined in terms of obtaining server SSL and personal certificates are explained in the following chapters.

6.1 The AMRES Server Certificate Issuance Service

In order for an institution to obtain a server SSL certificate, it needs to complete the following steps:

1. Register the institution at *ac.rs*,
2. Apply for the AMRES certificate issuance service,
3. Create a pair of keys and the certificate request,
4. Submit the certificate request.

The installation of certificates and the configuration of servers are described in section 6 Certificate Installation.

6.1.1 Registration of an Institution

The registration is performed through the *ac.rs* Domain Registry Portal.

An institution is considered registered with *ac.rs* if it has at least one domain registered with the ac.rs Domain Registry Portal. The registration of an institution's domain at *ac.rs* (whether the domain is already active or completely new) is performed only once. The domain of the institution thus becomes official in administrative terms in *ac.rs*, as the domain of an institution that is part of the academic and research community, which is automatically qualified to use other services of the Academic Network. The registered institution can, *inter alia*, apply to use the services of the AMRES TCS.

The data on the institution on the ac.rs Domain Registry Portal must be accurate and up to date. This data is used for verifying the information on the institution in the state registries of companies, in the document the institution signs and attaches to its application for using the AMRES TCS, and later in the process of verifying the information contained in the certificate request submitted by the institution in the process of obtaining a TCS certificate.

6.1.2 Application for Using the AMRES Server Certificate Issuance Service

The institutions registered at *ac.rs* can apply for using the AMRES Server Certificate Issuance Service by uploading the *Agreement on Registration with the Digital Server Certificate Issuance Service* filled in and signed by the authorised person via the portal of the *ac.rs* Domain Registry. By signing the Agreement, the institution appoints a person who will act as its administrative contact in the procedures for requesting, obtaining, renewing and revoking digital certificates. The terms, rights and obligations concerning the use of digital certificates specify that each institution, i.e. its appointed administrative contact, needs to be familiar with the basic preconditions for the use of digital certificates as defined by the *Rules Concerning the Use of the Server Certificate Issuance Service*. Following the verification of data by AMRES, the appointed administrative contact will be notified by e-mail about the outcome of the application.

The institution that becomes a user of the AMRES Server Certificate Issuance Service is further entitled to submit the request and obtain server SSL certificates through its administrative contact.

6.1.3 Creating an Asymmetric Key Pair and the Certificate Signing Request

Following successful registration with the AMRES Server Certificate Issuance Service, the next step is creating and submitting a request for obtaining a server SSL certificate in the form of a Certificate Signing Request (CSR).

The creation of the CSR request is preceded by the generation of an asymmetric pair of RSA keys, i.e. a private key and a corresponding public key, using the tools available on the server. On Linux servers, one should use the OpenSSL package (it can be found at http://www.openssl.org). On Microsoft platforms, the corresponding package is available in the group of Internet (Information) Service Manager tools. Once the keys are generated by means of these tools, a Server Certificate, i.e. the CSR request can be created.

Note: In order to achieve better security, AMRES adopted a rule according to which the minimum length of an asymmetric key pair generated in the process of creating the CSR request is 2048 bits.

The Server Certificate links the public key to the identity of the server, which makes it necessary to also include information that describe the server's identity in the CSR request as well as the public key. This primarily relates to the registered DNS name of the server (included in the *Common name* field of the certificate). If several DNS names (e.g. for different services) are used on the same physical machine (server), one of the registered DNS names needs to be designated as the primary DNS name of the server. All other names covered by the certificate are listed as additional/alternative DNS names of the server.

The Server Certificate that protects a server can contain one or more (maximum 100) of its DNS names. Each DNS name of the server (either primary or alternative) needs to be indicated in the FQDN form (Fully Qualified Domain Name), i.e. the full DNS name of the server must be provided (e.g. *www.sf.bg.ac.rs*, *www.uns.rs.ac*, etc.).

As explained in Chapter 4, a Single SSL Certificate covers only one DNS name of the server, while a Multi-Domain SSL Certificate covers several DNS names of the same server. Accordingly, a Multi-Domain SSL Certificate should be requested when the institution has several services on the same server, where each of these services uses its own DNS name. For instance, a server with the primary name 'main_server.at_domain.ac.rs, which includes the web service, webmail service, mail service, and pop3 service, accessed via www.at_domain.ac.rs, webmail.at_domain.ac.rs, mail.at_dommain.ac.rs and pop3.at_domain.ac.rs.

The specific procedure for creating CSR requests depends on the characteristics of the server for which the SSL certificate is envisaged (operating system, the tools available for creating the request on that platform, and the number of different DNS names for the given server). The procedure for creating a request on Linux servers (with OpenSSL package), the most commonly used platform at AMRES, is described below. The instructions for other server platforms can be found at: http://www.instantssl.com/ssl-certificate-support/csr_generation/ssl-certificate-index.html.

6.1.3.1 *Linux OpenSSL*

When generating a CSR request on Linux servers, one should use the OpenSSL package. The package should be downloaded from http://www.openssl.org and installed on the server.

Generating requests for Single DV SSL certificates

As explained in Chapter 4, a Single SSL Certificate covers only one DNS name of the server. A *SCSreq.cnf* configuration file has been prepared for generating this type of certificates.

SCSreq.cnf – the configuration file used in the process of generating a certificate request that contains one DNS name of the server (in the FQDN form, e.g. rcub.bg.ac.rs):

```
[ req ]
default_bits = 2048
default_keyfile = keyfile.pem
distinguished_name = req_distinguished_name
encrypt_key = no
```

```
[ req_distinguished_name ]
commonName = DNS(FQDN)name of the server
commonName_max = 64
```

The content of this *SCSreq.cnf* configuration file needs to be pasted into a text file on the server and saved under a corresponding name.

Note: the data should not be typed into the SCSreq.cnf configuration file; it is necessary to enter this data when activating the OpenSSL command (as explained below).

The certificate request is generated by activating the following commands:

```
umask 0377
openssl req -new -config SCSreq.cnf -utf8-keyout myserver.key -out
server.csr
```

The umask command should be used to define the read privileges for everything that will be created subsequently. The purpose of this is to protect the server private key, which should remain secret and should not be publicly available.

In activating the openssl command (the OpenSSL package needs to be previously installed), the name of the SCSreq.cnf configuration file is indicated (along with the path that leads to it, if it is located in a different directory to the one from which the command is activated), as well as the name of the file in which the private key will be located - myserver.key, and the name of the file in which the Certificate Signing Request will be kept (alternatively, we use the short name 'certificate request') - server.csr. The names myserver.key and server.csr can be changed, so it is convenient to name them after the server for which the request is generated.

By activating the openssl command, one enters into the dialogue shown in Figure 6.1. The command will request appropriate information concerning each of the questions. The default answers will be offered in square brackets [], if they are predefined. The requested data should be entered accurately and it should correspond to the data submitted in the registration process (e.g. use the official name of the institution under which it has been registered on the portal, etc.).

```
Generating a 2048 bit RSA private key
......+++
.........................+++
writing new private key to 'myserver.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
```

Country code (2 characters) [RS]:
Location name (city) [ ]: Belgrade
Institution name [University of Belgrade]:
Organisational unit name [RCUB]:
FQDN server name []: servername@rcub.bg.ac.rs

Figure 6.1:Request for information dialogue

Note: If some of the entries contain Latin characters such as č, ć, š, ž, đ, UTF-8 support should be activated. The set up depends on the terminal that is used. For SecureCRT terminal users (version 5.5.3.536), the set up should be performed in six steps in the order shown in Figure 6.2.

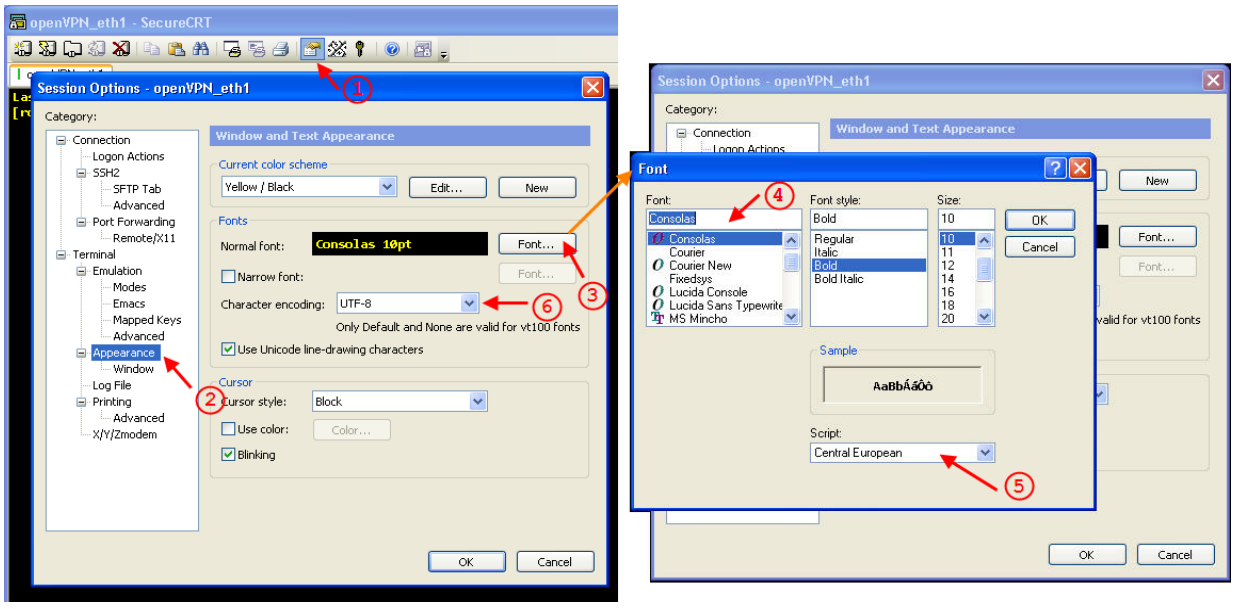

Figure 6.2: Setting up the UTF-8 support at SecureCRT terminals

Following the successful execution of the command, the Certificate Signing Request will be created (in the *server.csr* file), together with a pair of keys (of which the private key is in the *myserver.key* file, while the public key is in the *server.csr* file packed together with the request).

The content of the generated Certificate Signing Request can be verified using the following command (Figure 6.3):

```
openssl req -in server.csr -text
```

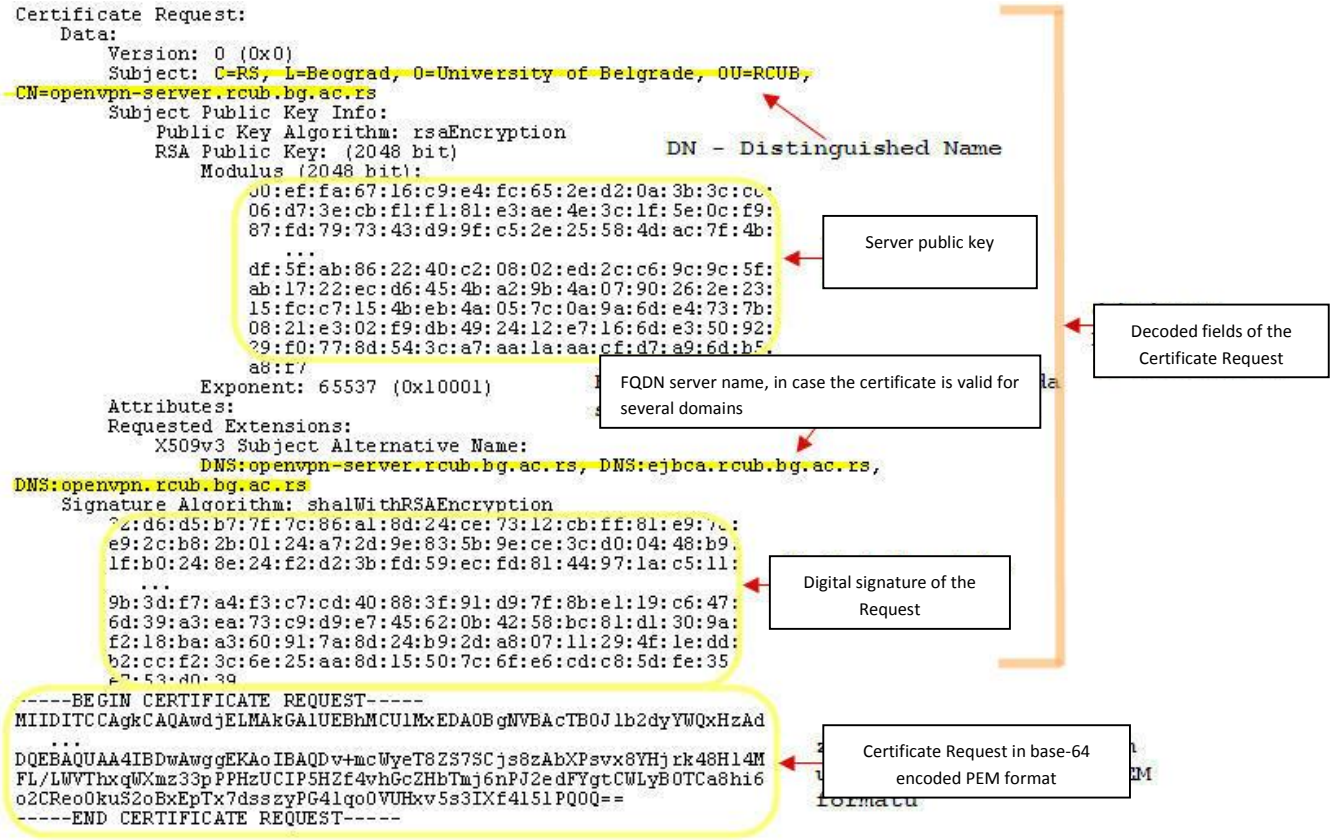

Figure 6.3: Certificate Request

All the data that should be included in the certificate, which is signed by the server private key, is entered in the appropriate format between "-----**BEGIN CERTIFICATE REQUEST-----**" **and** "**-----END CERTIFICATE REQUEST-----**", at the end of the file (*server.csr*).

The private key in the *myserver.key* file remains on the server and must not be publicly available. Recommendations concerning the selection and protection of directories in which certificates and keys are stored are provided in section 6, Certificate Installation.

Below is a description of the request generation procedure for other types of certificates. Only the commands that need to be executed in order to obtain the certificate request are provided. The explanation provided in this chapter also applies to the others steps of the procedure.

Generating requests for Multi-Domain DV SSL certificates

A Multi-Domain SSL Certificate covers more than one DNS name of the same server. It is very common to have multiple services with different symbolic addresses located on one Linux server (one IP address). For example, when several websites or services are hosted on one server, different DNS names of sites/services are mapped in the same IP address of the server. It would be wrong to request a separate server certificate for each website or service. Only one certificate that is valid for all the DNS names of the server should be requested in that case, specifically the Multi-Domain SSL Certificate.

A *MultiSCSreq.cnf* configuration file has been prepared for the purpose of generating requests for this type of certificates.

MultiSCSreq.cnf – the configuration file used in the process of generating a certificate request that contains more than one DNS name of the server:

```
[ req ]
default_bits = 2048
default_keyfile = keyfile.pem
distinguished_name = req_distinguished_name
encrypt_key = no
req_extensions = v3_req

[ req_distinguished_name ]
0.commonName          = Primary DNS (FQDN) name of the server
0.commonName_max            = 64

[ v3_req ]
subjectAltName = @alt_names

[ alt_names ]
DNS.1                              = www.at_domain.ac.rs
DNS.2                              = webmail.at_domain.ac.rs
DNS.3                              = pop3.at_domain.ac.rs
```

The content of the *MultiSCSreq.cnf* configuration file should be moved to a text file on the server and saved under a corresponding name. The name of the file is stated when prompted by OpenSSL to create the CSR request. When a certificate request that contains more than one DNS name is generated, all other DNS names need to be entered directly into the *MultiSCSreq.cnf* configuration file in the [alt_names] section as values of the DNS.x elements (x=1,2,3, ...). The given configuration file is formed for three additional DNS names, though this number can be lower or higher depending on need. If the number of DNS names is less than three, the excess DNS.x elements should be deleted (delete the entire line that begins with DNS.x).

The certificate request is generated using the following commands:

```
umask 0377
openssl req -new -config MultiSCSreq.cnf -utf8 -keyout myserver.key -out
server.csr
```

Generating requests for Wildcard DV SSL certificates

The procedure for generating a Wildcard DV SSL certificate request is the same as the procedure for generating a Single Domain DV SSL certificate request, except that the character * needs to be entered in front of the domain for which the certificate is requested, e.g. **.inst.bg.ac.rs*.

Generating requests for Single SSL OV certificates

A *SCSreqOV.cnf* configuration file is prepared for generating a request for this type of certificate.

SCSreqOV.cnf – the configuration file relied on when creating a certificate request containing one DNS server name, the name of the country (C=country), the name of the organisation (O=organisation) and the name of the organisational unit within which the certificate will be used (OU=organisational unit fields):

```
[ req ]
default_bits = 2048
default_keyfile = keyfile.pem
distinguished_name = req_distinguished_name
encrypt_key = no

[ req_distinguished_name ]
countryName = Country code (2 characters)
countryName_default = RS
countryName_min = 2
countryName_max = 2
localityName = Location name (city)
organizationName = Full name of the institution
organizationalUnitName = Name of the organisational unit
commonName      = DNS(FQDN)name of the server
commonName_max = 64
```

The content of this *SCSreqOV.conf* configuration file should be pasted into a text file on the server and saved under a corresponding name.

Note: the data should not be typed into the SCSreqOV.cnf configuration file; it is necessary to enter this data when activating the OpenSSL command (as explained below)

The certificate request is generated using the following commands:

```
umask 0377
openssl req -new -config SCSreqOV.cnf -utf8-keyout myserver.key -out
server.csr
```

Generating requests for Multi-Domain OV SSL certificates

A *MultiSCSreqOV.cnf* file is prepared for generating this type of certificate.

MultiSCSreqOV.cnf – the configuration file relied on when creating a certificate request containing more than one DNS server name, the name of the country (C=country), the name of the organisation (O=organisation) and the name of the organisational unit within which the certificate will be used (OU= organisational unit fields):

```
[ req ]
default_bits = 2048
default_keyfile = keyfile.pem
distinguished_name = req_distinguished_name
encrypt_key = no
req_extensions = v3_req

[ req_distinguished_name ]
countryName = Country code (2 characters)
countryName_default = RS
countryName_min = 2
countryName_max = 2
localityName = Location name (city)
organizationName = Full name of the institution
organizationalUnitName = Name of the organisational unit
commonName      = Primary DNS(FQDN)name of the server
commonName_max = 64

[ v3_req ]
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = www.at_domain.ac.rs
DNS.2     = webmail.at_domain.ac.rs
DNS.3     = pop3.at_domain.ac.rs
```

The content of the *MultiSCSreqOV.cnf* configuration file should be pasted into a text file on the server and saved under a corresponding name. The name of the file is stated when prompted by OpenSSL to create the CSR request.

When the certificate request that contains more than one DNS name is generated, all other DNS names need to be entered directly into the *MultiSCSreqOV.cnf* configuration file in the [alt_names] section as values of the DNS.x elements (x=1,2,3, ...). The given configuration file is formed for three additional DNS names, though this number can be lower or higher depending on need. If the number of additional DNS names is less than three, the excess DNS.x elements should be deleted (delete the entire line that begins with DNS.x).

The certificate request is generated using the following commands:

```
umask 0377
openssl req -new -config MultiSCSreqOV.cnf -utf8-keyout myserver.key -out
server.csr
```

Generating requests for Wildcard OV SSL certificates

The procedure for generating a Wildcard OV SSL certificate request is the same as the procedure for generating a Single Domain OV SSL certificate request, except that the character * needs to be entered in front of the domain for which the certificate is requested, e.g. **.inst.bg.ac.rs*.

Generating requests for SHA-2 certificates

In some cases it will be necessary to define a hash algorithm for signing the certificate in the certificate request. Below is the set of commands used for that purpose. The commands are provided using the example of generating a request for a Single Domain SSL certificate:

```
umask 0377
openssl req -new -sha256 -config SCSreq.cnf -utf8 -keyout myserver.key -
out server.csr
```

6.1.4 Submitting the Request

A certificate request can be submitted in two different ways:

- By sending an e-mail message with the certificate request to the authorised AMRES TCS administrator.

The Certificate Request is submitted by an institution's authorised person (administrative contact), on behalf of the AMRES Service user institution, who is designated in the process of registering the institution and submitting the application for using the service. The request is submitted by e-mail to the following address: tcs@amres.ac.rs. The content between the lines "-----BEGIN CERTIFICATE REQUEST-----" and "-----END CERTIFICATE REQUEST-----" from the file in which the Certificate Signing Request is located should be copied to the body of the message in text format. The Certificate Signing Request is in the base-64 encoded PEM format between the lines "-----BEGIN CERTIFICATE REQUEST-----" and "-----END CERTIFICATE REQUEST-----", including these lines.

An example of the content of the *myserver.csr* file is shown below:

```
-----BEGIN CERTIFICATE REQUEST-----
MIIDITCCAgkCAQAwdjELMAkGA1UEBhMCUlMxEDAOBgNVBAcTB0Jlb2dyYWQxHzAd
BgNVBAoTFlVuaXZlcnNpdHkgb2YgQmVsZ3JhZGUxDTALBgNVBAsTBFJDVUIxJTAj
BgNVBAMTHG9wZW52cG4tc2VydmVyLnJjdWIuYmcuYWMucnMwggEiMA0GCSqGSIb3
DQEBAQUAA4IBDwAwggEKAoIBAQDv+mcWyeT8ZS7SCjs8zAbXPsvx8YHjrk48H14M
+Yf9eXND2Z/FLiVYTax/S59YuFKi1vlmkxEFusspaDCnPs8dQovX2UYHZt9tNGXS
fzk2x7rviI/mG1y3o15Y0QH96Ov6+R2aGBAPcimjLtWh17KaAE0Xon4V6QWExNU6
0TkP73/krflXTehJh2GdT7OvPCbJnwXUTN/RxLqETyL/BlbQr0mmi7Kqdy3xQLJM
ng5kBQ+fkd9fq4YiQMIIAu0sxpycX6sXIuzWRUuim0oHkCYuIxX8xxVL60oFfAqa
OiQyonjCG7ZJXngh7OtESeJEEtCniy+fzzweedv62kLTjMx5Osxw6FzUMuXCugzg
8kDH3DS607iv4Gn3IhYk/aV6H6hJtwUZaA/vssst6MvM6SJOeePZbpvGbYbnUAXU
FL/LWVThxqWXmz33pPPHzUCIP5HZf4vhGcZHbTmj6nPJ2edFYgtCWLyB0TCa8hi6
o2CReo0kuS2oBxEpTx7dsszyPG4lqo0VUHxv5s3IXf4151PQOQ==
-----END CERTIFICATE REQUEST-----
```

The following data should also be included in the e-mail sent to tcs@amres.ac.rs:

- full official name of the institution under which it has been registered at the portal of the ac.rs Domain Registry (https://registar.ac.rs), and the address of the institution, i.e. street and street number, postcode, and the name of the city in which the institution is based;
- all the DNS names of the server covered by the certificate contained in the Certificate Signing Request (if multiple symbolic names are specified in one certificate, indicate the primary name);
- the validity period of the certificate for which the request is submitted – one, two or three years;
- the server software used for generating the request;
- the data concerning the authorised person / administrative contact of the institution submitting the request;
- the e-mail address to which the signed certificate will be sent.

After AMRES has successfully verified the data contained in the request, the certificate will be issued and sent to the e-mail address specified in the Certificate Request.

- By sending the Certificate Request via the central portal.

Certificates can also be requested through the central portal, which automates the process of issuing certificates and enables the authorised administrators and institutions to send their certificate requests directly. Currently, AMRES uses the AMRES TCS Portal based on the DjangoRA open-source application developed by the SUNET (Swedish University Computer Network) and Linköping University. The administrative and technical contacts of the AMRES TCS user institutions can request certificates through the AMRES TCS Portal, in accordance with the information available on the *ac.rs* Domain Registry Portal.

6.2 AMRES Personal Certificate Issuance Service

The AMRES TCS Service also offers the issuance of personal certificates. Unlike the Server Certificate Issuance Service, this service is offered to end users of the AMRES user institutions. In this case, TERENA also has the role of the Certification Authority. AMRES regulates the process of issuing certificates, while the AMRES user institutions are responsible for maintaining databases of their users and act as registration authorities, i.e. they decide which end users will have the right to use this Service.

In order for an institution to ensure the services of obtaining personal certificates for its users, it needs to complete the following steps:

1. Registering the institution at *ac.rs*,
2. Becoming a member of the iAMRES identity federation,
3. Applying for the AMRES Personal Certificate Issuance Service,

4. User registration.

After the institution has successfully joined this Service, each of its users must pass the following:

5. The certificate issuance procedure.

6.2.1 Registering the Institution

An institution is considered registered with *ac.rs* if it has at least one domain on the *ac.rs* Domain Registry Portal. The registration of an institution's domain at *ac.rs* (whether the domain is already active or completely new) is performed only once. The domain of the institution thus becomes official in administrative terms. The institutions that have their domains registered with *ac.rs* are, as institutions of the academic and research community, automatically qualified to use the other services of the Academic Network. The registered institution can, *inter alia*, apply to use the AMRES Personal Certificate Issuance Service (by signing the appropriate agreement and submitting it via the *ac.rs* Domain Registry Portal).

6.2.2 Becoming a Member of the iAMRES Identity Federation

In order to ensure the services of issuing personal certificates, an institution must become a member of the iAMRES Identity Federation. To that end, the institution must provide and maintain an up-to-date database of its users and ensure a connection to the AMRES RADIUS infrastructure.

6.2.3 Applying for the TCS Personal Certificate Issuance Service

The institutions registered at *ac.rs* can apply to use the AMRES Server Certificate Issuance Service by uploading the *Agreement on the Registration with the Personal Digital Certificate Issuance Service* filled in and signed by the authorised person via the *ac.rs* Domain Registry Portal. By signing the Agreement, the institution appoints a person who will act as its administrative contact in procedures related to the use of the Service. The terms, rights and obligations concerning the use of digital certificates specify that each institution, i.e. its appointed administrative contact, needs to be familiar with the basic preconditions for using digital certificates as defined by the *Rules Concerning the Use of the Personal Certificate Issuance Service*. Following the verification of data by AMRES, the appointed administrative contact will be notified by e-mail about the outcome of the application.

6.2.4 User Registration

For a user within the institution to use the Personal Certificate Issuance Service, he or she will have to pass the registration process. It is the institution that performs the registration of its users for the purpose of using the AMRES Personal Certificate Issuance Service. The registration process requires the user to personally present an official document issued by the competent authority (identification card, passport or driving licence) to the institution's administrator. The institution is responsible for

maintaining an internal database of users. The data in the institution's user database must be accurate and up to date.

The relevant data includes:

- The username, which must be unique and have the following format: user_name@domain_institution.ac.rs. The username is entered in the *Subject* field of the certificate.
- The full name of the institution, entered in the *Subject* field of the certificate.
- The e-mail address of the user, which must be in the *ac.rs* domain. One user can have one or more e-mail addresses.
- The full name of the user, which must match the name in the official document issued by the competent authority (identification card, passport or driving licence). The full name of the user is entered in the *Subject* field of the certificate.
- The access permission attribute, which comprises a set of characters defining the right of the user to use the Personal Certificate Issuance Service. The attribute is assigned to the user account in the institution's user database following the successful registration of the user for this Service.

6.2.5 The Certificate Issuance Procedure

The certificate issuance procedure is regulated by AMRES. For this purpose, AMRES has provided a central portal that automates the procedure for issuing certificates to the users of institutions that are members of this Service. The current solution is based on the Confusa open-source application developed by the UNINETT Academic Network of Norway and the Nordic DataGrid Federation.

7 Certificate Installation

During the installation of the obtained server certificate, it is recommended to configure the server in such a way that along with its own SSL certificate, it also sends the *ca-chain* file establishing the chain of trust for that certificate. The *ca-chain* file contains the intermediate and root certificates. While most of the SSL clients have the most commonly used root certificates already preinstalled, in order for each server SSL certificate to be verified, the client also needs information on the intermediate certificate signed by the CA root certificate.

The procedure for installing server SSL certificates for the most commonly used services/servers at AMRES – the certificates for protection of services/servers on the Linux platform (with Apache/mod_ssl package), is described below.

If an institution has requested a Single SSL Certificate, the procedure for the relevant environment should be followed (exclusively) based on the instructions contained in one of the following subchapters.

If an institution has requested a Multi-Domain SSL Certificate, which means that the institution uses a platform that comprises multiple services on the same server (where each of the services uses its own DNS name), all the subchapters relating to individual services should be followed.

7.1 Web Server under Linux (Apache/mod_ssl)

The Server Certificate can be found in a file with the *.crt* or *.pem* extension, while the chain of certificates for establishing a chain of trust for the issued Server Certificate is located in a file with the *.ca-bundle* or *.pem* extension. It is assumed here that the Server Certificate is contained in the *cert-9026687.pem* file, the chain of trust is in *chain-9026687.pem*, and the private key is located in the *myserver.key* file. The instructions given below also apply if files with other extensions are used. In the case of Linux distributions derived from Red Hat (CentOS, Fedora, Mandriva), certificates and keys are usually stored in the following locations, respectively:

```
/etc/pki/tls/certs/
/etc/pki/tls/private/
```

You should edit the *ssl.conf* configuration file and enter the appropriate paths to the Server Certificate, the chain of certificates and the server key:

```
#(path to the server SSL certificate)
SSLCertificateFile /etc/pki/tls/certs/cert-9026687.pem
#(path to the server private key)
SSLCertificateKeyFile /etc/pki/tls/private/myserver.key

#(path to the intermediate certificate)
SSLCertificateChainFile /etc/pki/tls/certs/chain-9026687.pem
```

In the end, you should restart the web server:

```
/etc/init.d/httpd stop
/etc/init.d/httpd start
```

If you want to install a Multi-Domain SSL Certificate, you should go to the next service that is protected by the certificate and enter the data on the path to the directory in which the certificates are stored in the corresponding configuration file.

7.2 Java Web Server (Tomcat, JBoss, etc.)

In order to use the obtained certificate on the Tomcat Java web server, it is necessary to import the certificate and its private key into the so-called *Java keystore* file. This file contains the private key and the certificate used by Java for SSL communication.

First, it is necessary to create a file in the *PKCS #12* format that should contain both the private key and the certificate, which is done using the following command:

```
cd /home/tomcat/
openssl pkcs12 -export -chain -in cert-9026687.pem -inkey myserver.key -
CAfile chain-9026687.pem > server.p12
```

where *cert-9026687.pem* is the path to the certificate in the *.pem* format, *myserver.key* is the path to the private key of the certificate, *chain-9026687* is the path to the *ca-chain* file containing the interim and root certificate, and *server.p12* is the generated *keystore* file.

Then, it is necessary to import *server.p12* into the Java keystore:

```
Keytool –importkeystore –srckeystore server.p12 –destkeystore server.jks -
srcstoretype pkcs12
```

Note: the password to be used when creating the server.jks file is: "changeit".

The next step is configuring Tomcat so that it can establish HTTPS communication. It is necessary to activate the HTTPS connector in the *conf/server.xml* Tomcat file (which already exists, but contains comments). The HTTPS connector should look as shown below:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
      maxThreads="150" scheme="https" secure="true"
      clientAuth="false"                                    sslProtocol="TLS"
keystoreFile="/home/tomcat/server.jks" />
```

Make sure you add `keystoreFile="/home/tomcat/server.jks"` , which defines the path to the *keystore* file used for this purpose.

7.3 Radius Server

This chapter describes the procedure for installing and using a server SSL certificate on a RADIUS server (built on the FreeRadius platform), which uses the EAP-TTLS protocol for user authentication. This environment is commonly used in AMRES when the RADIUS server of a member institution is included in *eduroam* service.

The obtained certificate and chain of trust should be placed in the `/etc/raddb/certs` directory, which usually stores certificates related to the FreeRadius server. This directory is also used for storing the server private key generated in the process of requesting the given server certificate. It is assumed here that the private key is kept in the *myserver.key* file, the server is in the *cert-9026687* file and the chain of trust is in the *chain-9026687.pem* file.

If the certificate is provided in the *.crt* format; it should be converted into the *.pem* format using the following commands:

```
openssl x509 -in cert-9026687.crt -out cert-9026687.der -outform DER
openssl x509 -in cert-9026687.der -inform DER -out cert-9026687.pem -outform
PEM
```

Access to the server private key needs to be limited, which is ensured by the “r--------“ (read only) permission. The “r--------“ (read only) permission can be activated by using the following command:

```
chmod 400 /etc/raddb/certs/myserver.key
```

In order for the authentication to function through the *ttls* protocol, you should first configure the *tls* in the *eap.conf* configuration file. Edit the *eap.conf* configuration file and enter the information on the certificates by using the following commands:

```
certdir = /etc/raddb/certs
```

```
cadir = /etc/raddb/certs
private_key_file = /etc/raddb/certs/myserver.key
certificate_file = /etc/raddb/certs/cert-9026687.pem
CA_file = /etc/raddb/certs/ chain-9026687.pem
```

Now you need to restart the RADIUS process in order for the changes in the configuration to take effect:

```
killall radiusd
radiusd
```

In order for a client to successfully verify the Server Certificate, the client needs to install the certificate on its computer.

7.4 E-mail on a Linux Server

The procedure for installing a certificate for the e-mail service on a Linux server is similar to the process of installing a certificate for the web server described in Chapter 6.1.1 Web Server under Linux (Apache/mod_ssl), with the difference that different configuration files are configured. The appropriate configuration file is selected depending on the specific e-mail software used for exchanging e-mails. As a general rule, the paths to the SSL Server and Intermediate Certificates, as well as to the private key (corresponding to the public key contained in the certificate) need to be entered in each configuration file of the service secured with the certificate.

The following commands are relevant for a Linux server on which a *postfix* package is installed in order to enable the exchange of e-mails based on the ***smtp*** protocol and a *dovecot* package for downloading messages via *imap* and *pop3* protocols. It is assumed that the private key is kept in the *myserver.key* file, the server is in the *cert-9026687.pem* file, and the chain of trust is in the *chain-9026687.pem* file.

- For the Postfix SMTP server, you should edit the *main.cf* configuration file (*/etc/postfix/main.cf*):

```
smtpd_use_tls = yes

#(path to the intermediate certificate)
smtpd_tls_CAfile = /etc/pki/tls/certs/ chain-9026687.pem

#(path to the directory containing certificates)
smtpd_tls_CApath = /etc/pki/tls/certs

#(path to the server certificate)
smtpd_tls_cert_file = /etc/pki/tls/certs/ cert-9026687.pem

#(path to the private key)
```

```
smtpd_tls_key_file = /etc/pki/tls/private/ myserver.key
```

- For IMAP and POP3 protocols on the server, you should edit the *dovecot.conf* configuration file *(/etc/dovecot.conf):*

```
#(path to the certificate server)
ssl_cert_file= /etc/pki/tls/certs/cert-9026687.pem

#(path to the server private key)
ssl_key_file = /etc/pki/tls/private/myserver.key

#(path to the intermediate certificate)
ssl_ca_file = /etc/pki/tls/certs/ chain-9026687.pem
```

When installing a Multi-Domain SSL certificate, you should move to the next service protected by the certificate and insert the path to the directory in which the certificates are kept in its corresponding configuration file.

References

[1]	William Stallings, “Network and Internetwork Security: Principles and Practice”, Prentice Hall, 1995
[2]	Pascal Steichen, “PKI applications, standards and protocols” http://pst.libre.lu/mssi-luxmbg/p3/02_std-prot-art.html
[3]	http://www.itu.int/rec/T-REC-X.509-200508-I
[4]	https://support.comodo.com/
[5]	http://tools.ietf.org/html/rfc2595
[6]	http://www.rfc-editor.org/rfc/rfc2487.txt
[7]	http://en.wikipedia.org/wiki/Public_key_infrastructure
[8]	Stephen A. Thomas, “SSL and TLS essentials: Securing the Web”, Wiley, 2000
[9]	Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, “Handbook of Applied Cryptography” http://www.cacr.math.uwaterloo.ca/hac/
[10]	Portal of the ac.rs Domain Registry https://registar.ac.rs/DNS-web/pages/home.jsf

Glossary

CRL	Certificate Revocation List
FQDN	Fully Qualified Domain Name
IETF	Internet Engineering Task Force
IGTF	International Grid Trust Federation
NREN	National Research and Education Network
PKI	Public Key Infrastructure
TERENA	Trans-European Research and Education Networking Association (since October 2014, GÉANT Association)
TCS	TERENA Certificate Service
X.509 standard	ITU-T standard for the Public Key Infrastructure (PKI). The X.509 standard specifies, *inter alia*, the standard formats for digital certificates, the Certificate Revocation List and the certificate properties
RADIUS	Remote Authentication Dial In User Service

Complete BPDs are available at http://services.geant.net/cbp/Pages/Home.aspx

campus-bp-announcements@terena.org