Securing Service Access with Digital Certificates

Best Practice Document

Produced by AMRES Security Group
(AMRES BPD 106)

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

April 2011

Document No:	GN3-NA3-T4-AMRES-BDP-106
Version / date:	April 2011
Source language:	Serbian
Original title:	"*Upotrebom digitalnih sertifikata do sigurnog pristupa servisima*"
Original version / date:	Revision 1 (of the document dated September 2010) / 21 April 2011
Contact:	milica.kovinic@rcub.bg.ac.rs, dpajin@rcub.bg.ac.rs

AMRES/RCUB is responsible for the contents of this document. The document was developed by the Security Topic Group organised by AMRES with the purpose 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° 238875, relating to the project 'Multi-Gigabit European Research and Education Network and Associated Services (GN3)'.

Table of Contents

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 sent or received. Such security is provided by the use of digital certificates in conjunction with Secure Sockets Layer (SSL) technology.

The document outlines the components of a Public Key Infrastructure (PKI), and also the implementation of PKI functions to include AMRES in the TERENA Certificate Service (TCS). The document specifies various needs for PKI in a National Research and Education Networking organisation (NREN), which require various types of digital certificates, while special attention has been given to the use of PKI and digital certificates in combination with SSL technology for the purpose of the mutual authentication of services and their users.

The 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.

Introduction

As a result of the development of electronic communication, confidential information is exchanged on a daily basis. In order to receive or send sensitive data (e.g., usernames) when accessing web content, mail or RADIUS servers, users must be sure 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 Secure Sockets Layer (SSL) technology on servers (HTTPS, POP3S, etc.) provides the necessary security, and requires the servers to have the appropriate digital certificates.

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

The purpose of this document is to promote the use of server 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 Public Key Infrastructure (PKI). This part also describes the format and types of digital certificates for different purposes at an NREN (National Research and Education Networking organisation).

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.

1 Basic Terms Concerning Secure Communication and PKI

This document uses technical security terms concerning the use of the PKI 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 identified and explained below.

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

- **Authentication** is the process of establishing the identity of the end users in communication (e.g., by way of digital signatures).
- The **Integrity** of data guarantees that there has been no change to the data or the content of a message on its way from the source to the destination, most often by way of generating a hash of the message (a fixed-length sequence of 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 form a digital signature. When it is only the sender who has access to his/her private key, this means that only the sender could have sent the relevant digitally signed message.

Cryptographic keys include:

- **Asymmetric Key Pair** – 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 by the public key can only be read by the private key owner, while a message encrypted by the private key can be read by all the persons who know the public key.

- **Private Key** – mathematical data that can be used for creating an electronic signature or decrypting a document encrypted using the asymmetric cryptographic algorithm so that only the private key owner can access it. 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 the verification of 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 by way of a 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 by means of encryption system concepts using asymmetric algorithms. The following definitions are used in relation to PKI:

- **Certification Authority (CA)** – refers to a legal entity that issues digital certificates.
- **Registration Authority (RA)** – refers to an entity responsible for the initial identification and verification of data about 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** – refers to an electronic document confirming that the public key belongs to a certain entity (RFC 5280). The term 'electronic certificate' is also used.
- **CA certificate** – refers to the certificate of the CA itself. It confirms that the CA is 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** – refers to an ordered sequence of certificates that is processed together with the public key of the initial object in the chain in order to be verified in the last object of the chain. It forms a hierarchical chain of trust, in which one digital certificate confirms the authenticity of the previous digital certificate.
- **Certificate Policy (CP)** – refers to a designated set of rules that defines the applicability of a certificate to an environment and/or a class of applications with common security requirements.
- **Certificate Practice Statement (CPS)** – refers to 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 checking the appropriate Certificate Revocation List (CRL).
- **Signatory** – refers to an entity that has a means for electronic signing and performs electronic signing in its own name. It can be an individual or a legal entity to whom a certificate has been issued.
- **User (end entity, certificate owner)** – is an individual or a legal entity to whom a digital certificate is issued. Individuals are end users, while legal entities are institutions that request digital certificates for their facilities, most often for their servers or services.
- **Personal user certificate (personal or client certificate)** – refers to an electronic certificate issued to an individual or an 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 his/her username within the identity federation, the full name of the user and e-mail address.

- **Server certificate – an electronic certificate issued to a legal entity** or an institution for the purpose of verifying the identity of a server or a service that it owns. It contains data that identifies the service, most often the Fully Qualified Domain Name of the server.
- **Certificate Service Request (CSR)** – refers to a standard format (according to recommendation PKCS #10) used for sending certificate service requests.
- **Certification** – refers to the process of issuing a digital certificate.
- **Certificate serial number** – refers to a number assigned to a certificate, which is unique for each certificate issued by a specific CA.
- **Certificate Revocation List (CRL)** – refers to 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** – refers to the permanent revocation of the validity of a specific certificate and its inclusion in the CRL.
- **Repository** – refers to 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)** – refer to a group of public key cryptography standards defined by the RSA Laboratories.

2 Cryptographic Protocols and Techniques

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

2.1 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 communication is achieved by way of encryption of the messages.

Figure 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, which also contribute to changes in 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 on the confidentiality of the algorithm. The message, modified (encrypted) in this manner, is then sent through a non-secure telecommunication channel towards its destination. The message could be intercepted by an interceptor without the risk of him gaining an advantage through learning the information contained in the message or changing the original message and sending false information.

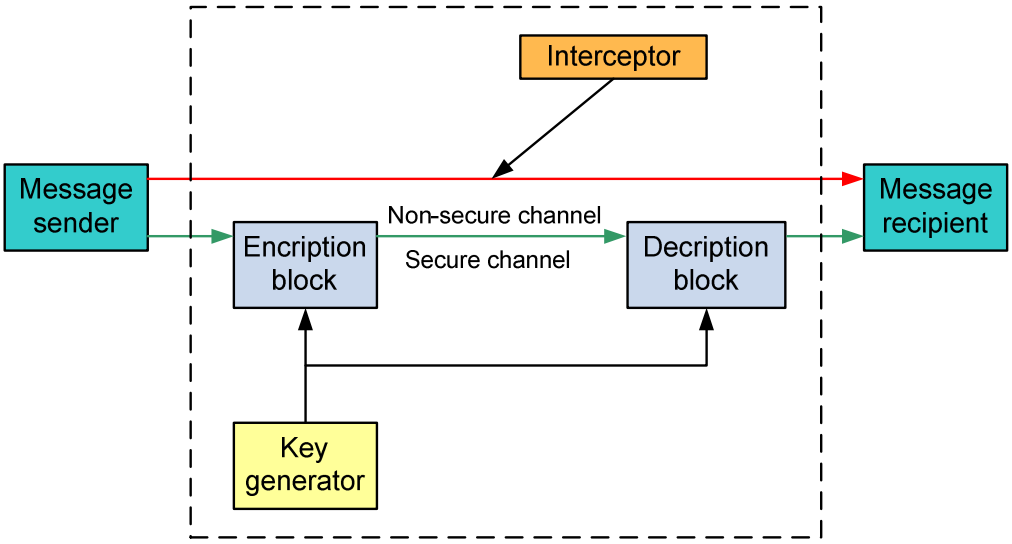

Figure 1: A general block-scheme of the encryption system

2.1.1 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 about 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 verification of the message integrity or the authentication of its sender.

2.1.2 Asymmetric-Key Encryption Systems

Attempts to resolve the problem of the scalability and distribution of keys between participants in communication have resulted in 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 to detect the private key based on the known, public key (within 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 by using the public key of the recipient. Thus, only the intended recipient can decrypt the message since he is the only one that has the corresponding private key.

In order to ensure authentication, the sender encrypts the data he sends by way of his private key. When the recipient receives the message, he decrypts it 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 digital signatures.

2.1.3 Combined Encryption Systems

The correct 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 keys of smaller sizes, while the scalability and distribution of keys are achieved by way of asymmetric cryptography algorithms, which are more complex and require more processing time.

Figure 2 shows a block-scheme of a combined encryption system.

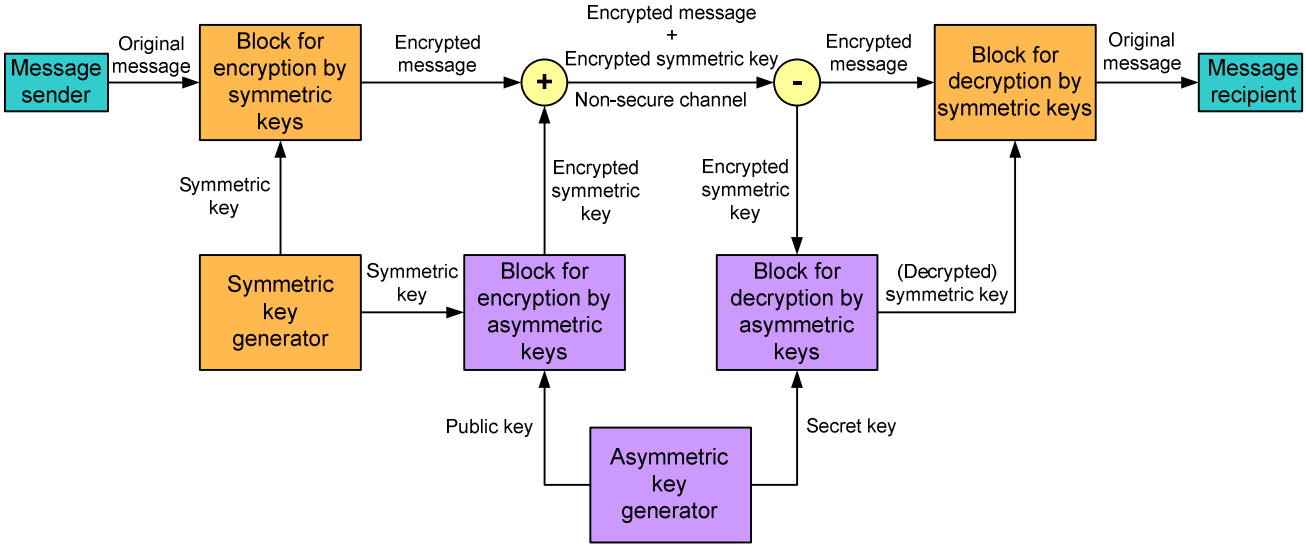

Figure 2: A block-scheme of a combined encryption system

2.2 Integrity Verification – Hash Functions

The recipient needs to be certain that the content of the message he receives 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 fixed-length sequence of bits, also known as the hash value of the message, or message digest, or simply digest. The basic characteristic of the hash function is that even the smallest change in the original message results in a change of 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 the given hash value. Algorithms that have this characteristic are called one-way algorithms;
- if a message is known, it is impossible (in a finite number of steps) to find another 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 1, below, contains a list of 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 provides better security.

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 1: Hash functions

2.3 Verifying the Integrity with Authentication – MAC

A Message Authentication Code (MAC) basically has the characteristics of a hash, with the addition of a shared key, which, besides the verification of the integrity of the message, also enables the authentication of the sender. Like a hash function, a MAC is a fixed-length sequence of bits obtained 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 the 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 (such as virus activity), while the message recipient can identify its sender.

There are two categories of MAC, 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 that he did not send the message or that someone else sent the message on his behalf. Digital certificates are used to ensure protection from non-repudiation.

2.4 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) by way of digital signatures.

The digital signature of electronic data can be viewed as analogous with 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, as well as to be certain of the identity of the message sender. In other words, it guarantees the integrity of the message and the identity/authentication of its sender. In addition, the digital signature cannot be denied, so that the person who sent the message cannot subsequently claim that he did not send it or that some other person sent it.

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 were even slightly changed, the resulting hash of the message would also be changed. The sender then encrypts the hash with his own private key. A 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 he then compares it with the value of

the hash he receives 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 claims to be.

Figure 3 shows a block-scheme of a system with the digital signature.

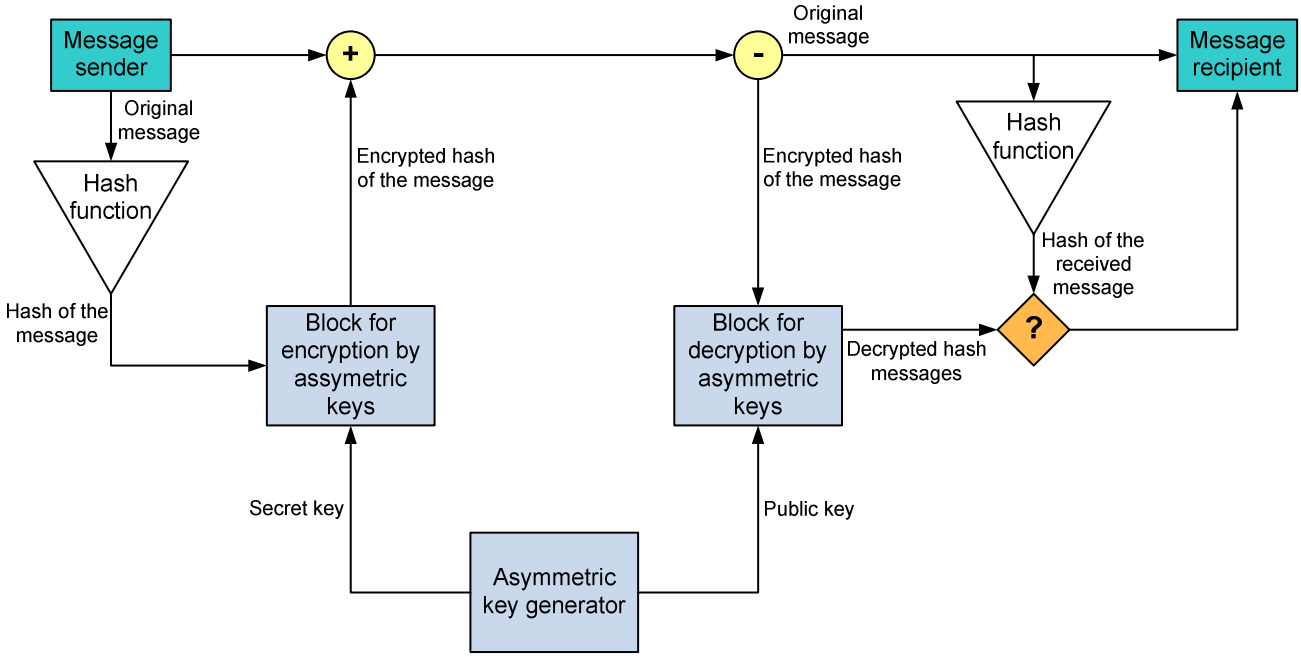

Figure 3: A block-scheme of a system with the digital signature

2.5 Digital Certificate and Public Key Infrastructure

In using the asymmetric cryptography technique, either to ensure the confidentiality of information by way of encryption or by way of 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 indeed belongs to the person we believe is its owner. A potential attacker can intercept and send his own public key in order to read the message intended for someone else or to compromise the integrity of the message by signing it with his own 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, but also an end server). This prevents the possibility that keys are 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 CA is responsible for issuing certificates and it represents 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 PKI 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 CA that issued the certificate), the serial number of the certificate and the digital signature of the issuer.

The most commonly used format of the 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 and JKS).

When sending a message, the sender digitally signs the message and sends his 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 hierarchical 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) CA, the Root Certification Authority, trusted by all other certification authorities. The public key of the Root Certification Authority needs to be publicly known and available. Appendix A presents a chain of trust based on the real example of the CA certificate chain for the TCS.

Figure 4 shows a block-scheme of a system with digital certificates.

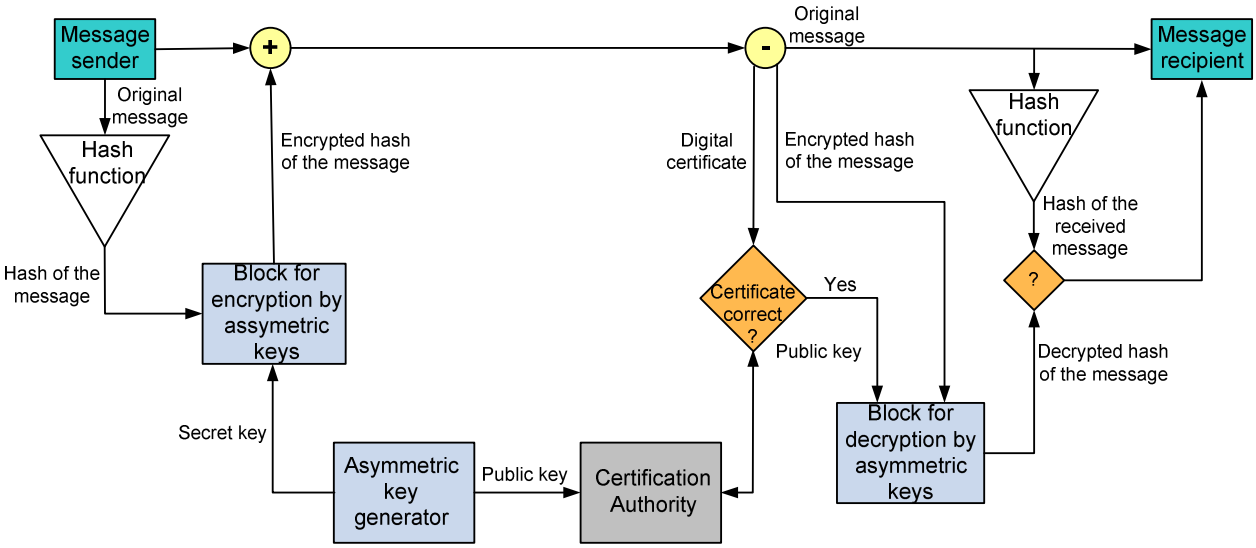

Figure 4: A block-scheme of a system with digital certificates

3 Public Key Infrastructure (PKI)

The Public Key Infrastructure (PKI) contains the 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 encryption-system concepts that use asymmetric algorithms.

PKI is defined by the ITU-T X.509 standard, together with IETF document RFC 5280, which provides more detailed recommendations for the Internet community.

3.1 PKI – Components and Basic Functions

The PKI is comprised of the hardware, software, policies and procedures needed to manage, generate, store, distribute, use and revoke cryptographic keys and digital certificates.

Figure 5 shows the relationship between the basic PKI elements.

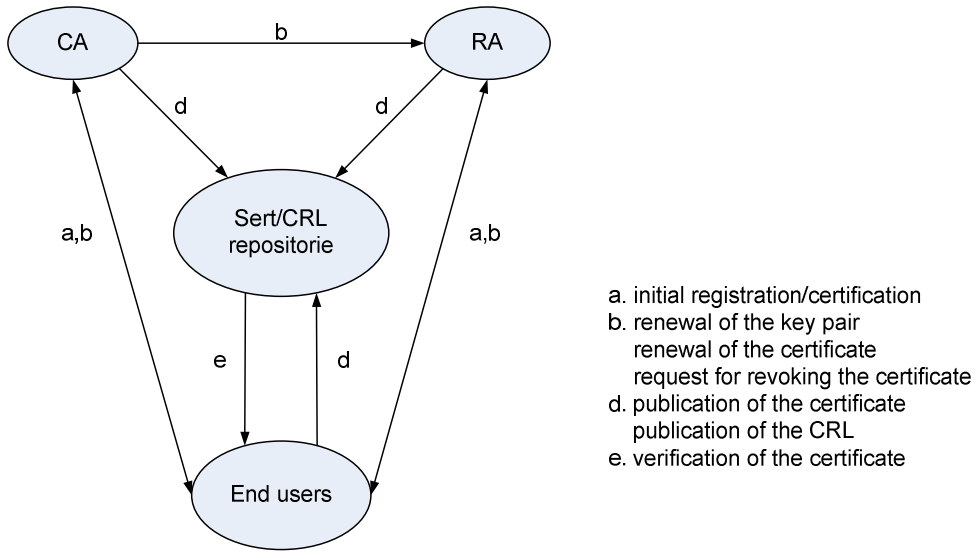

Figure 5: The relationship between the PKI elements

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 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 of 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 the data, the CA issues a digital certificate to its owner/end entity. This certificate guarantees that the public key belongs to that specific owner/end entity. The end entity can be an end user (an 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 his identity and it is signed by the digital certificate of the CA itself. End users now use their own digital certificates, issued by a trusted CA, in order to prove their identity to one another and to establish secure communication.

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

3.1.1 Registration, the Application Process for Institutions/Users

In order to use the services of the PKI, institutions/end users first need to go through a process of application that includes verification of their identity and 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 verification of the identity of an end entity that has submitted a certificate request. With respect to the end user, 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 this reason, the procedure for institutions also includes a pre-registration stage that involves verification of the data about an institution and the submission of the names of persons (as part of the procedure of the designation of the institution's representatives) that will be authorised to submit certificate requests on behalf of the institution. The level of verification of the identity of an end user or an institution depends upon 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**.

3.1.2 Initialisation

The next step in the registration process is initialisation. The representative of an institution/end user and the CA 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 CA, and the manner of generating and delivering an asymmetric key pair.

3.1.3 Certification

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

3.1.4 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, requiring 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 CA 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).

In some cases, real-time certificate status protocols are used instead of the CRLs, which provide positive or negative information about 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 may provide a chain of certificates, where each certificate is signed by the certificate of the superior CA. This requires verification of the chain of trust and the validity of each certificate contained in the chain.

3.1.5 Verification of the Chain of Trust

Verification of each individual certificate requires answers to these 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, he 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., the trusted CA that owns the certificate. Appendix A shows the chain of trust established using the real example of the chain of CA certificates for the TCS.

3.1.6 Verification of the Validity of a Certificate

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 CRLs and 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.

3.2 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 (Figure 6):

- **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** – the identifier in the OID (object identifier) format. It identifies the algorithm used by the CA for signing the certificate.

- **Issuer** – the unique name of the CA 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 point in time that the certificate becomes valid (the field *Not Valid Before)*, while the other indicates the point in time time until which the certificate can be considered valid (the field *Not Valid After)*.
- **Subject** – identifies the user/entity to which the certificate has been issued. Just as in 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 (such as, RSA, DSA or Diffie-Hellman).
- **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 about the type of extension, the critical/non-critical mark and the content of the extension. If the user software cannot recognise the extension during processing, and if the extension is marked as a critical extension, the certificate will be marked as invalid. (If the extension is marked as non-critical, it can be ignored and the offered certificate can be accepted.) The extensions in use belong to the following categories:
  - **Standard extensions:**
    - **Authority key identifier** (**AKI**) – enables the identification of the public key that corresponds to the private key of the CA that has signed the certificate. This is usually used when the CA has more than one key for signing.
    - **Subject key identifier** (**SKI**) – a unique identifier used for identifying the certificates that contain the corresponding public key.
    - **Key usage** – defines the purpose of the key contained in a 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 Object Identifier Format (OID), 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 of the Practice Statement of the Certification Authority) 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 that is to be specified in this extension is indicated in the *Issuer domain policy* and *Subject domain policy* fields.
    - **Subject alternative name** – contains additional data about the certificate subject/object (e.g., the end user's e-mail, URL, IP address, and alternative DNS name of the server).
    - **Issuer alternative name** – contains data about the CA that issued the certificate (e.g., e-mail, URI, and IP address).
    - **Subject directory attributes** – provides additional information on the certificate subject (e.g., nationality) and is not of great importance.
    - **Basic constraints** – provides information about whether the subject of the certificate is a CA or an end user/entity. In the case of a CA, the extension can also contain the *Path length* parameter (which shows the number of certificates of certification authorities that can follow it).

- **Name constraints** – used only in the certificates of CAs; provides space for 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 CAs must/must not belong.
- **Policy constraints** – used in the certificates of CAs in order to prevent the 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 indicates 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 indicates the usage of a certain certificate policy in a certain number (defined by this value) of certificates that follow.
- **Extended key usage** – used in the certificates of end users/entities to indicate additional possibilities of 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 the certificates of CAs to define the relationship of a special value of a certificate policy – *anyPolicy* (which specifies the possibility of using any certificate policy) with the following CAs. The value refers to the number of additional certificates that can appear in the chain before *anyPolicy* is no longer allowed, i.e., the number of CAs after this, before *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 to the Delta list of revoked certificates.

- **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 CA issuing a certificate (e.g., about the service for online verification of the certificate status, or additional information about the certificate policy).
  - **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 CA).

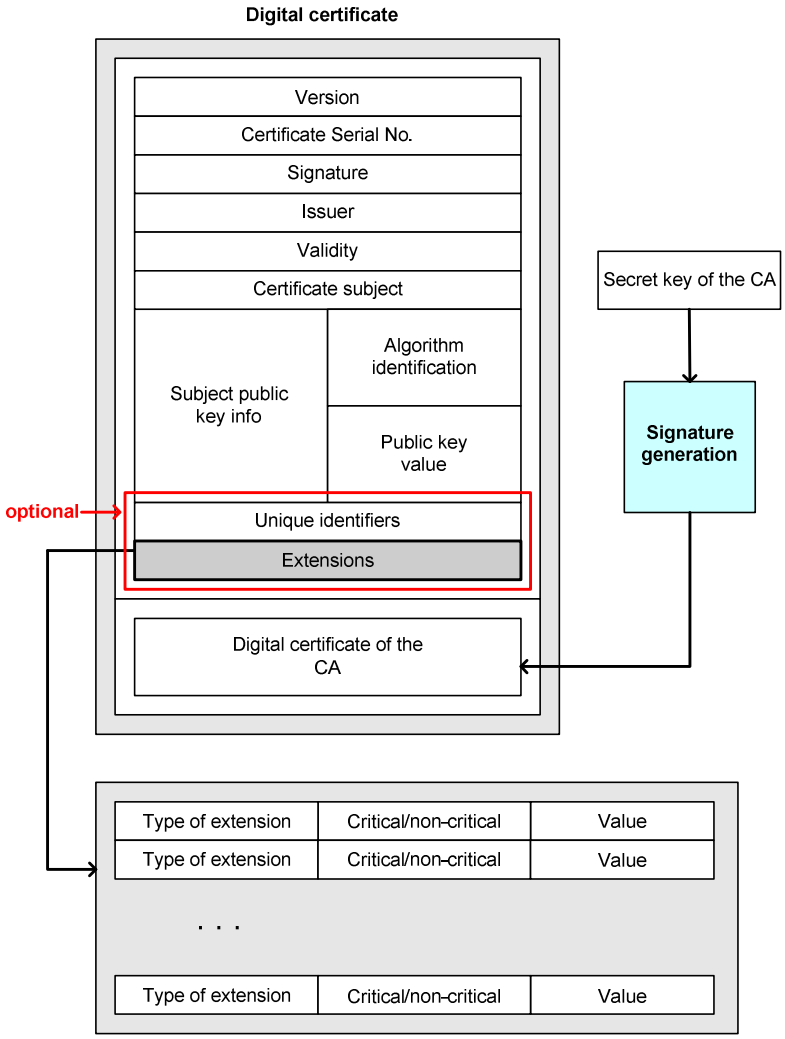

Figure 6: A digital certificate according to RFC 5280

Examples of formats and the real content of the fields contained in a digital certificate can be found in Appendix A, which shows the chain of the CA certificates for the TCS.

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

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).

4 The TCS – TERENA Certificate Service

The TCS (TERENA Certificate Service) issues digital certificates to scientific, research and education institutions through their National Research and Education Networking organisations (NRENs). TERENA has subcontracted the role of Certification Authority to a commercial provider, while the role of Registration Authority is entrusted to the national TERENA members (NRENs) that have subscribed to the TCS. The certificates obtained through the TCS are issued by Comodo CA Limited, one of the leading, globally recognised certification authorities.

4.1 Types of Certificate Offered by TCS

TCS offers five different types of digital certificates:

- **Server SSL Certificate (Server Certificate)** – an SSL certificate 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:
  - **TERENA Single SSL Certificate** – this type of certificate is linked to only one registered DNS name of the server, which is included in the certificate as the value in the CN (*Common Name*) attribute.
  - **TERENA Multi-Domain SSL Certificate** – this type of certificate secures 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 in the SAN (*Subject Alternative Name*) field. This certificate can contain up to 100 different DNS names of services located on **one** physical machine (server).
  - **TERENA Wildcard SSL Certificate** – one certificate allows for an unlimited number of subdomains located on different physical machines (servers). This certificate contains the domain name in the form *.name_of_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 Wildcard Certificate is only justified in the following cases:
    - web server farms
    - clusters
    - load balancers
    - failover servers.

- **e-Science Server Certificate** – for authenticating Grid hosts and services. They are IGTF compliant. The validity period of these certificates is up to 13 months.

- **Personal Certificate** – also called Client Certificate, is used for authenticating or identifying end users accessing user servers/services and securing e-mail communications (the digital signing of e-mails and encryption of their content). These certificates contain data that identifies the end user, such as the username within the identity federation (the establishment of the AMRES identity federation is ongoing), user's full name, e-mail address, etc. The validity period of these certificates is up to three years.

- **e-Science Personal Certificate** – for authenticating or identifying end users accessing Grid services. They are IGTF compliant. The validity period of these certificates is up to 13 months.

- **Code-signing Certificate** – for authenticating software distributed over the Internet. The validity period of these certificates is up to five years.

4.2 The Advantages of TCS Certificates

The certificates obtained through the TCS are signed by the TERENA CA certificate, which is further signed by UserTrust, an intermediate CA, which in turn is signed by the AddTrust External Root CA. The root certificate is pre-installed in most of the SSL clients. For example, when implementing https access to a website that 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. The corresponding Root CA certificate is already pre-installed in most of the commonly used web browsers: Internet Explorer, Mozilla Firefox, Google Chrome and Opera.

Appendix A shows the chain of CA certificates for TCS server certificates. The first certificate in the sequence is the certificate of the Root CA – AddTrust External CA Root, by which the certificate of the intermediate CA – UTN-USERFirst-Hardware, is signed. At the end, there is a TERENA CA certificate – TERENA SSL CA - signed by the intermediate certificate mentioned above.

The AMRES TCS certificate service issues all types of TCS certificates to its member institutions, with the exception of Code-Signing and software authentication certificates. However, it should be noted that the certificate policy requires that clear and efficient procedures for data verification are established to allow issuing and using certificates. These procedures are currently established for server SSL certificates. AMRES plans to regulate the issuance of Personal Certificates in the same manner in the near future through the AMRES identity federation, which is currently being developed.

Server SSL certificates can be used on all AMRES servers that are not part of the Grid infrastructure. The issuance of e-science types 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 document that can be found at http://aegis-ca.rcub.bg.ac.rs/documents/AEGIS-CP-CPSv1-2.doc, which also contains the Certificate Practice Statement of the AEGIS CA.

4.3 Services that Need to Be Secured with Digital Certificates

Digital certificates should be used for the following purposes:

- **Authenticating servers** (web, mail, etc.) towards 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 that he has accessed the right server, which is important if sensitive data such as user credentials are sent. In addition to the authentication of servers, these certificates can also 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.
- **Authenticating end users, clients** towards 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 his Personal Certificate used to authenticate him to access Grid services.
- **The secure exchange of e-mails,** ensuring the preservation of the integrity and confidentiality of communication. The sender digitally signs an e-mail message with his own private key, which guarantees the integrity of the message. In order to ensure confidentiality, it is necessary that the participants in communication exchange their Personal Certificates in advance.
- **Establishing an IPsec/TLS VPN tunnel** and the authentication of end users who have initiated the establishment of the VPN tunnel. In addition, in the case of SSL VPNs, the certificates are also used for authentication of the SSL VPN server by VPN clients. Therefore, a SSL certificate and Personal Certificates are required in this case.
- **Digitally signing software** ensures proof of origin and the integrity of the software code. When the software code is digitally signed with a certificate issued by a trusted CA, the user can be certain that it has not been maliciously modified and that it is safe to install it on his system. This requires a Code Signing Certificate.

Taking into account that the use of digital certificates in member institutions of AMRES is still in its initial phase, the recommendations concerning the commonly used server/services that need to be secured with server SSL certificates are provided below. The protection of e-mail by using Personal Certificates is also explained.

4.3.1 Web Server

The HTTPS Protocol (Hypertext Transfer Protocol Secure) is used in order to ensure secure communication between a web server and a client through an unprotected network, such as the Internet. Secure communication, established in this manner, enables the client to authenticate the server it accesses, as well as the encryption of the 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 uses the SSL certificate of the server. 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 and enters 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 CA of the web server. If the certificate is found, the

client can trust the CA that issued the certificate. The client then 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 encrypts 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 by 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 a CA trusted by the client, a pop-up message appears (Figure 7) with a warning for the user that the site he is accessing is not secure, and questions whether the user wants to continue further communication (which can often confuse the user).

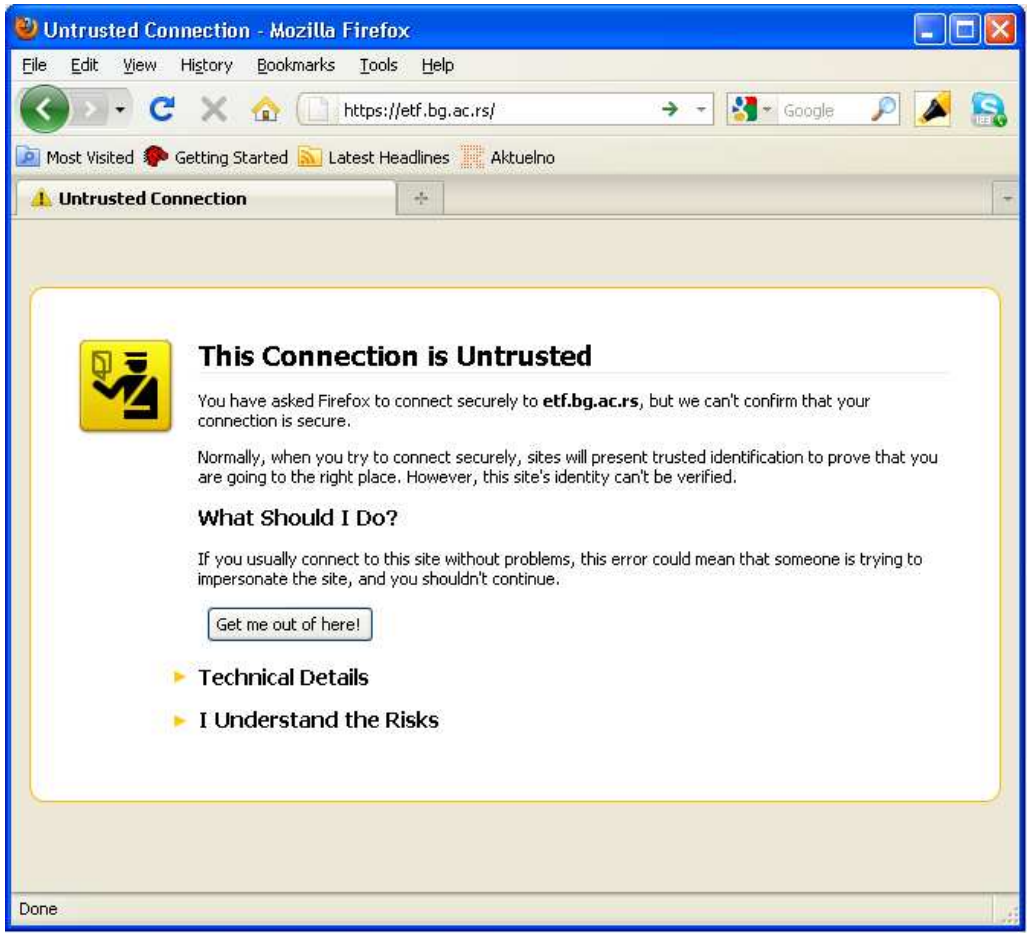

Figure 7: A message warning the user that the website he is accessing is not safe

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.

4.3.2 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*® service is

mandatory. The IEEE 802.1x standard is used for authenticating clients. EAP (Extensible Authentication Protocol) is used for conveying authentication messages in combination with TLS, TTLS (Tunnelled TLS) or PEAP (Protected EAP) protocols.

In the authentication process, the RADIUS server receives a message from the client that marks the beginning of the authentication. The server responds by sending its digital certificate to the client. The client verifies the certificate by trying to find the certificate of the CA that issued the Server Certificate in its local database. If the outcome is positive, the client sends its credentials (identity and password) encrypted using the public keys of the server. The RADIUS server decrypts the received data using the private key and verifies the user's credentials in its database.

A more detailed explanation of the EAP-TTLS Protocol can be found in Appendix C.

4.3.3 E-mail Server

STARTTLS is an extension of the SMTP, IMAP and POP3 protocols (SMTPS, IMAPS and POP3S) that enables the establishment an encrypted connection with the support of the SSL/TLS Protocol. Since STARTTLS is an upgrade of existing protocols, no special port is required for encrypted communication. Separate ports are registered for the SMTPS, IMAPS and POP3S protocols, although the RFC does not recommend them to be used 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 to mark the exchange of protected data.

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

In the case of encrypted communication between a client and a 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 with its private key.

It should be noted here that protected communication is not performed from one end to the other, but only between the 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 further on in the text.

4.3.4 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 that ensures the protected exchange of e-mails by enabling the authentication of the sender, proof of 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 8 shows the S/MIME encryption/decryption of a message. In order to encrypt a message, the sender generates a symmetric key for that session, with which he encrypts the body of the message. The sender then encrypts the symmetric key using the public key of the recipient. Upon receipt, the recipient uses his private key to decrypt the symmetric key generated for that session, which he then uses to decrypt the body of the message. This results in the confidential exchange of an e-mail message, in which the entire process is only fully transparent for the end users.

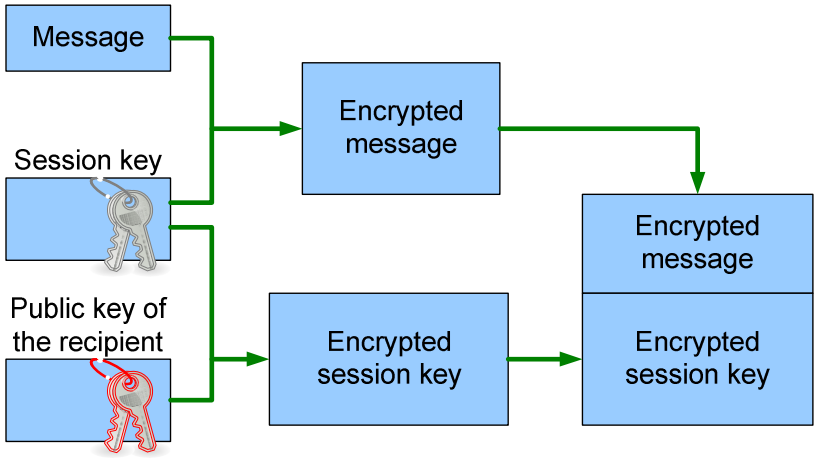

Figure 8: S/MIME encryption/decryption of a message

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 decryption of 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.

5 The AMRES TCS Certificate Service

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

The Certificate Policy prescribes the procedures for verifying data in the process of the issuance and use of certificates. The procedures for obtaining server SSL certificates are explained below.

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 to use the TCS;
3. create a pair of keys and the certificate request,
4. submit the request.

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

5.1 Registering an Institution

Registration is performed through the ac.rs Domain Registry portal at https://registar.ac.rs/DNS-web/pages/home.jsf .

An institution is considered registered with the *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 at *ac.rs*, as the domain of an institution that is part of the academic and research community, and which is automatically qualified to use other services of the academic network. The registered institution can, apply for using the TCS (by signing the appropriate agreement and submitting it via the portal).

The data about the institution must be accurate and up-to-date. This data is used for verifying the information about the institution in the state registries of companies, in the document the institution signs and attaches to its

application for using the TCS service, and later, in the process of verifying the information contained in the certificate request submitted to AMRES by the institution in the process of obtaining a TCS certificate.

5.2 Application for Using the TCS

The institutions registered at *ac.rs* can apply to use the TCS by uploading the TCS Terms of Use Agreement, filled in and signed by the authorised person via the portal of the *ac.rs* Registry (Figure 9). The agreement document can be downloaded from the AMRES website. When signing the document, the institution appoints a person who will act as its administrative contact for the procedures of requesting, obtaining, renewing and revoking digital certificates. The terms, rights and obligations concerning the use of digital certificates (defined by TERENA) specify that each institution, i.e., its appointed administrative contact, needs to be familiar with the basic preconditions for the use of digital certificates (which can be downloaded from the AMRES website). Following the verification of the data by AMRES, the appointed administrative contact will be notified by e-mail about the outcome of the application.

Figure 9: Sending the registration agreement for the TCS via Portal

The institution that becomes a user of the TCS is subsequently entitled to submit a request and obtain server SSL certificates through its administrative contact.

5.3 Creating an Asymmetric Key Pair and a Certificate Signing Request

The request for obtaining a server SSL certificate is submitted in the form of a Certificate Signing Request (CSR) sent by e-mail to: tcs@amres.ac.rs.

The creation of the CSR is preceded by the procedure of generating 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, the OpenSSL package should be used (this 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 have been generated using these tools, a Server Certificate, i.e., the CSR can be created.

Note: In order to achieve better security, AMRES has adopted a rule requiring that the minimum length of an asymmetric key pair, generated in the process of creation of the CSR, needs to be a minimum of 2048 bits.

The Server Certificate links the public key to the identity of the server, i.e., the DNS record about the server, so it is necessary to include in the CSR information that describes the server's identity, in addition to the public key. This primarily relates to the registered DNS name of the server (that will be 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 (a maximum of 100) 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*).

As explained in Chapter 4, the TERENA Single SSL Certificate covers only one DNS name of the server, while the TERENA Multi-Domain SSL Certificate covers several DNS names on the same server. Accordingly, a TERENA Multi-Domain SSL Certificate should be requested when the institution has several services on the same server, with each of these services using its own DNS name. For instance, the server with the primary name 'main_server.at_domain.ac.rs'provides *web, webmail, mail* and *pop3* services, via the DNS names: 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 a CSR depends on the characteristics of the server for which the SSL certificate is intended (operating system, 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 the two most commonly used platforms in AMRES – Linux servers (with the OpenSSLl package) and Microsoft IIS servers (4.x, 5.x/6.x) – is described below. The instructions for other server platforms can be found at www.instantssl.com/ssl-certificate-support/csr_generation/ssl-certificate-index.html.

5.3.1 Linux OpenSSL

In generating a CSR on Linux servers, the OpenSSL package should be used. The package should be downloaded from www.openssl.org and installed on the server.

Two predefined OpenSSL configuration files - SCSreq.cnf and MultiSCSreq.cnf - have been prepared for the two most common situations. One of these files should be selected and used depending whether a TERENA Single SSL Certificate (*SCSreq.cnf*) request or a TERENA Multi-Domain SSL Certificate (*MultiSCSreq.cnf*) request is submitted.

As explained in Chapter 4, a TERENA Single SSL Certificate covers only one DNS name for the server, while a TERENA Multi-Domain SSL Certificate covers several DNS names for the same server. Usually, Linux servers are used in such a way that several services with different symbolic addresses are 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, valid for all the DNS names on the server, should be requested in that case, specifically the TERENA Multi-Domain SSL Certificate (*MultiSCSreq.cnf*).

The content of the selected configuration file – SCSreq.cnf or MultiSCSreq.cnf – should be converted into a text file on the server and saved under an appropriate name. The name of the file is indicated in the OpenSSL invitation to generate a CSR. In this example, the same names are used, i.e., SCSreq.cnf for generating the certificate request that contains one DNS name of the server, and MultiSCSreq.cnf for generating a request for several DNS names.

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 ]
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
```

MultiSCSreq.cnf – the configuration file used in the process of generating a certificate request that contains more than one DNS name on 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 ]
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
0.commonName                    = Primary DNS (FQDN) name of the server
0.commonName_max                = 54

[ 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
```

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, but this number, depending on the needs, can be lower or higher. 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 by activating the *openssl* command (the OpenSSL package needs to be previously installed).

Before activating the *openssl* command, the *umask* command should be used to define the *read* privileges for everything that will be created subsequently. This is to protect the server private key, which should remain secret and should not be publicly available.

In activating the *openssl* command, the name of the configuration file is indicated (along with the path to it, if it is located in a directory other than the one from which the command is activated), as well as the name of the file in which the private key will be located and the name of the file in which the Certificate Signing Request will be located (alternatively, the short name ‘certificate request’ can be used). In the command presented below, the *SCSreq.cnf* configuration file is used, while *myserver.key* and *server.csr* are the names of the files where the server private key and the certificate request will be located, respectively. The names *myserver.key* and *server.csr* can be changed and it is convenient to give them the name of the server for which the request is generated.

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

By activating the *openssl* command, the dialogue shown in Figure 10 is entered. 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 correctly 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).

```
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
Full name of the institution []:University of Belgrade
Name of the organisational unit []:RCUB
DNS(FQDN)name of the server []:servername.rcub.bg.ac.rs
```

Figure 10

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 11.

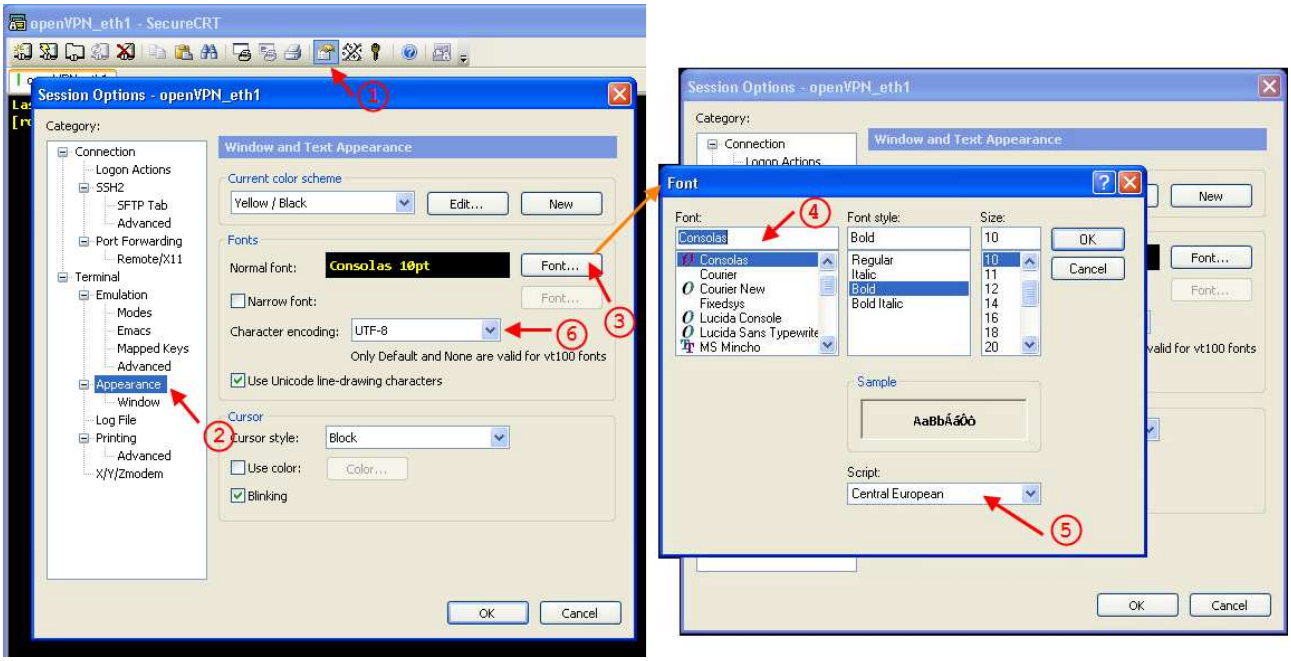

Figure 11: Setting up UTF-8 support for SecureCRT terminals

Following the successful execution of the command, the Certificate Signing Request is received (in the *server.csr* file) as well as 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 by using the following command (Figure 12):

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

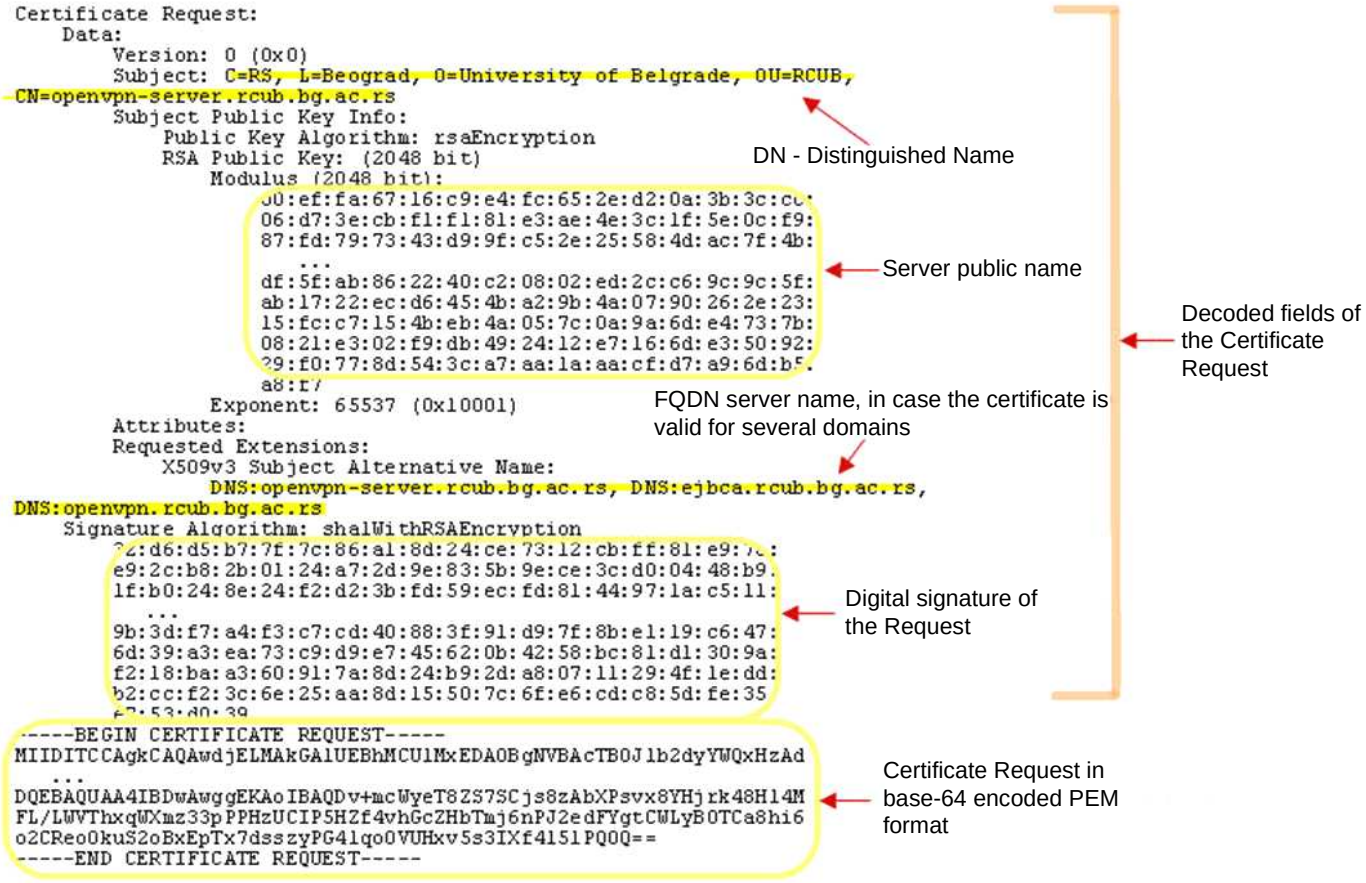

Figure 12: Certificate Request

It can be seen that besides the server public key, the Certificate Signing Request contains other data that should be included in a certificate, such as data on the institution, the primary DNS name, and optionally, several additional DNS server names. 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 it must not be publicly available. Recommendations concerning the selection and protection of directories in which certificates and keys are stored are provided in Chapter 6: Certificate Installation.

5.3.2 Microsoft IIS 4.x

For the Microsoft platforms, only the procedure for requesting a TERENA Single SSL Certificate for one DNS server name is indicated, and not the procedure of requesting a TERENA Multi-Domain SSL Certificate, which covers more than one DNS name of the same server (see Sections 4.1: Types of Certificates Offered by TCS and 5.3.1: Linux OpenSSL). Unlike Linux servers, Microsoft IIS servers in AMRES are rarely used for running multiple services with different symbolic addresses on one IIS server (one IP address).

The process of generating a key pair and the IIS SSL Certificate Signing Request (CSR) comprises the following steps:

- Open the **Microsoft Management Console (MMC)** for IIS (Option Pack > Microsoft Internet Information Server > Internet Service Manager).
- In the MMC, expand the **Internet Information Server** folder and select the **Computer name**.
- Open the **Properties** window for the website the CSR is created for by right-clicking on the website name.
- Open **Directory Security Folder**.
- In the **Secure Communications** area, select the **Key Manager** button and then select "**Create New Key...**".
- Choose "**Put the request in a file that you will send to an authority**". Enter an appropriate name (or accept the default name) for the file in which the generated request will be located.
- Fill in the appropriate details:
  - Fill in all the fields without using the following characters: **! @ # $ % ^ * ( ) ~ ? > < & / \;**

Note: For the sake of better security, AMRES has adopted a rule which requires that the minimum length of an asymmetric key pair generated in the process of creating the CSR request needs to be a minimum of 2048 bits.

However, if the IIS server is 40-bit enabled, the length of the generated keys must not exceed 512 bits; if the server is 128-bit-enabled, the length of the key can be up to 2048 bits. An exception to the general 2048 bits minimum rule can be made for 40-bit servers, although the level of protection is lower in that case.

- Click **Next** as many times as necessary, then click **Finish**.
- The **Key Manager** will display a key icon under the *www* icon. The key will have an orange slash through it indicating that it is not complete.
- In the **Computers** menu, select **Exit**. Select **YES** when asked to confirm the changes made.
- The obtained Certificate Request is in the file (whose name you have chosen in one of the previous steps) between the lines "**-----BEGIN CERTIFICATE REQUEST-----**" and "**-----END CERTIFICATE REQUEST-----** ", including these lines.
- Click **Next**.

Note: The password that protects the private key, as well as the key itself, should be known only to the institution's administrative contact.

5.3.3 Microsoft IIS 5.x / 6.x

For the Microsoft platforms, only the procedure for requesting a TERENA Single SSL Certificate for one DNS server name is indicated, not the procedure for requesting a TERENA Multi-Domain SSL Certificate, which covers more than one DNS name of the same server (see Sections 4.1: Types of Certificates Offered by TCS and 5.3.1: Linux OpenSSL). Unlike Linux servers, Microsoft IIS servers in AMRES are rarely used for running multiple services with different symbolic addresses on one IIS server (one IP address).

Note: Before generating the request, *hotfix* needs to be installed, which can be downloaded from http://support.microsoft.com/kb/ 75701; it resolves the problem of the country code for Serbia. Due to an error in the Web Server Certificate Wizard, the country code for Serbia is SP instead of RS.

The process of generating a key pair and the IIS SSL Certificate Signing Request (CSR) comprises the following steps:

- Select **Administrative Tools** and start the **Internet Services Manager**.

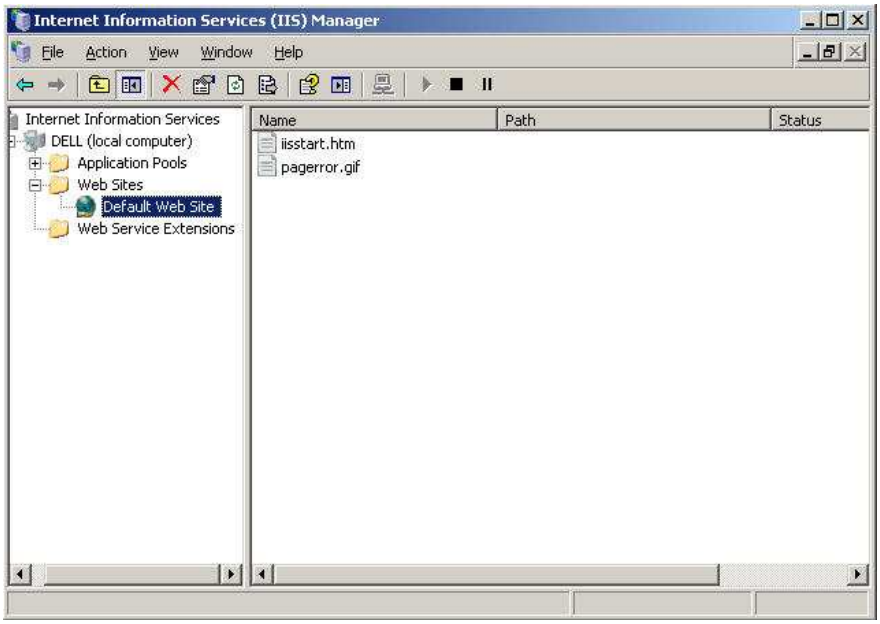

Figure 13

- Open the **Properties** window for the website for which the CSR is generated (by right-clicking on the **Default Website** and selecting Properties from the menu).

- Open **Directory Security** by right-clicking on the **Directory Security** tab.

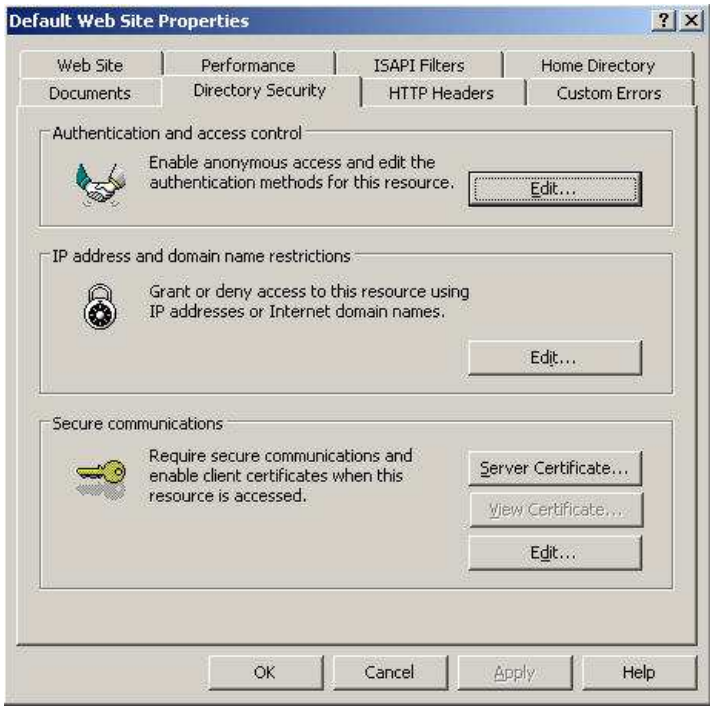

Figure 14

- By clicking on **Server Certificate**, the following Wizard will appear:

Figure 15

- Select **Create a new certificate**, then click **Next**.

Note: If the screenshot is not the same as the screenshot shown on Figure 16, which allows creation of a new request, the site most probably already has an SSL certificate. In this case, the procedure that enables the creation of a new certificate signing request should be applied, without removing the existing certificate. The instructions for this procedure can be found at

https://support.comodo.com/index.php?_m=knowledgebase&_a=viewarticle&kbarticleid=456.

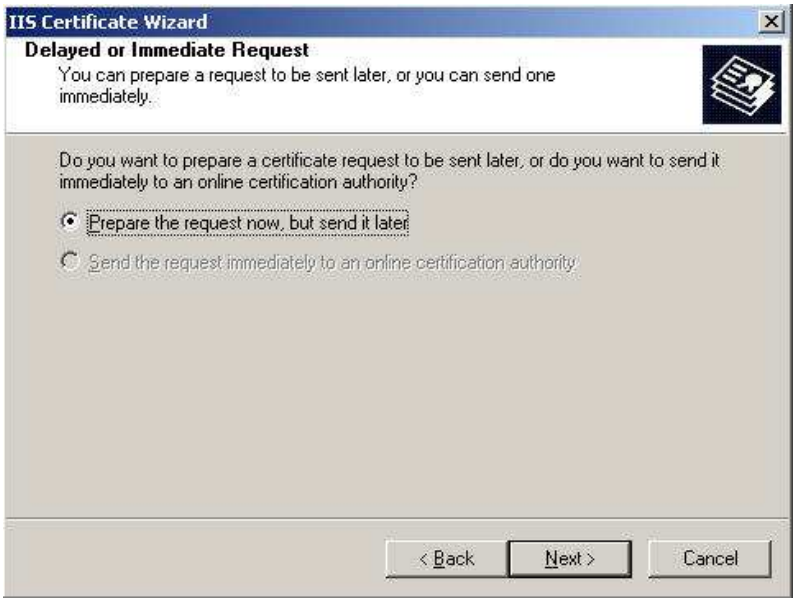

Figure 16

- Select **Prepare the request** and click **Next**.

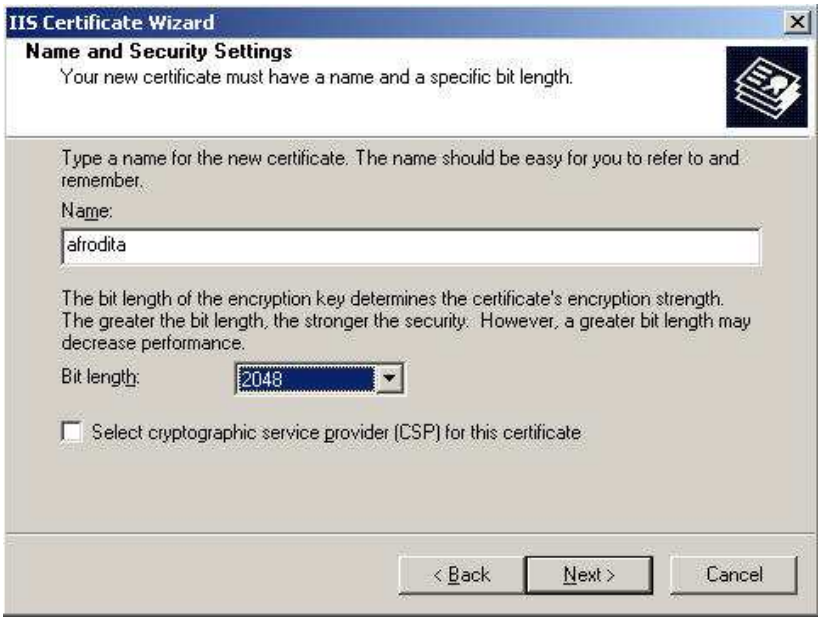

Figure 17

- Provide a name for the certificate. The name only has local meaning and should point to the specific certificate, if multiple certificates are in use on a domain.

- For **Bit length,** select **2048**, then click **Next**.

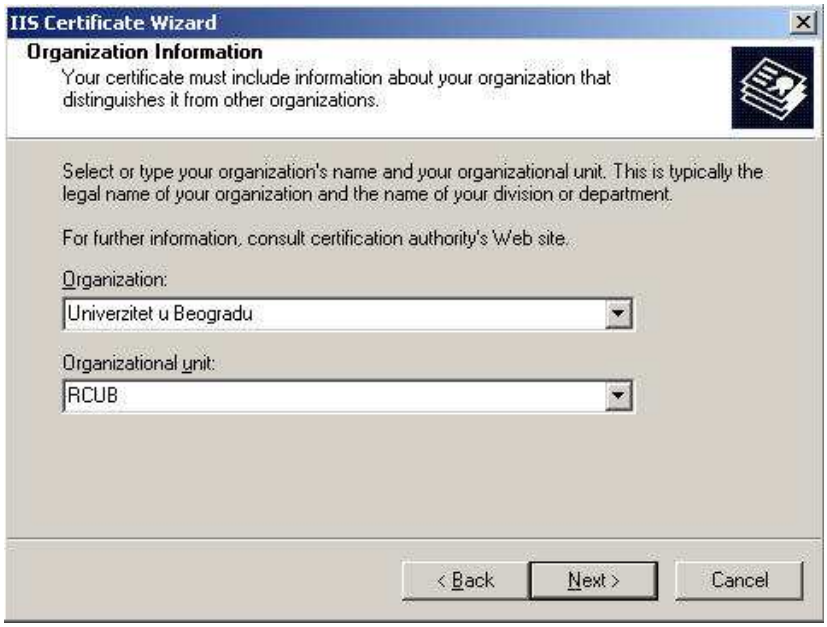

Figure 18

- Enter **Organization** (the full name of the institution) and **Organizational Unit** (the name of the organisational unit). From this point onwards, all the requested data needs be entered accurately and in accordance with the data submitted in the registration process (e.g., use the official name of the institution under which it has been registered at the portal).

- Click **Next**.

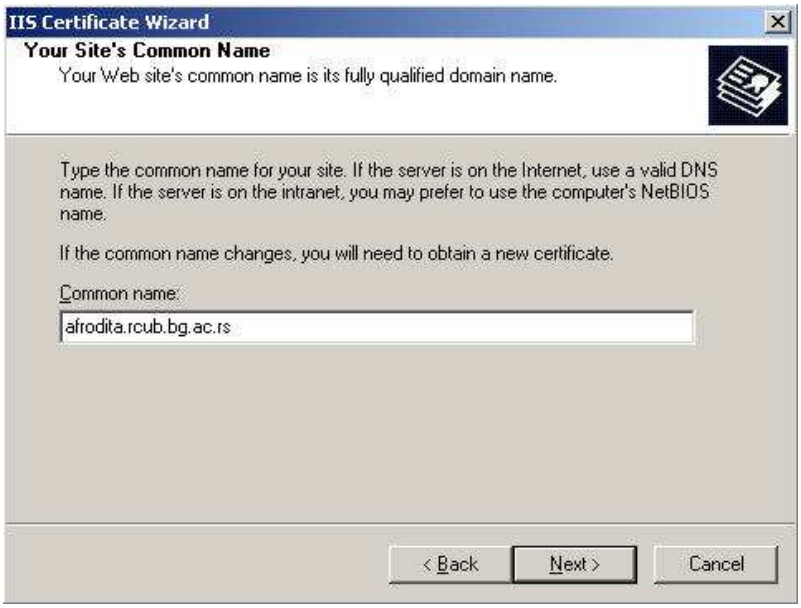

Figure 19

- The Fully Qualified Domain Name for which the certificate is requested should be entered in the **Common Name** field. Click **Next**.

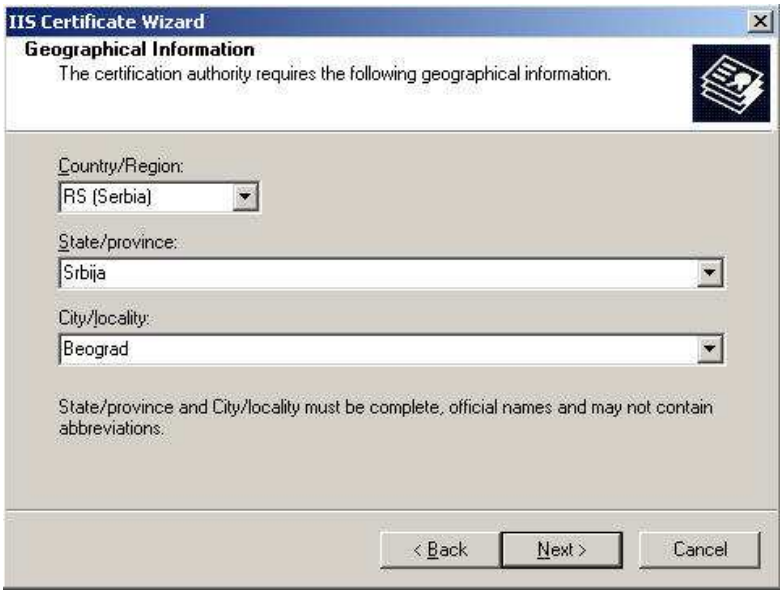

Figure 20

- Enter the two-character abbreviation for Serbia, state and city in the fields **Country**, **State** and **City**.

- Click **Next**.

Figure 21

- Enter a filename (use the *txt* extension) and location to save the Certificate Request. Click **Next**.

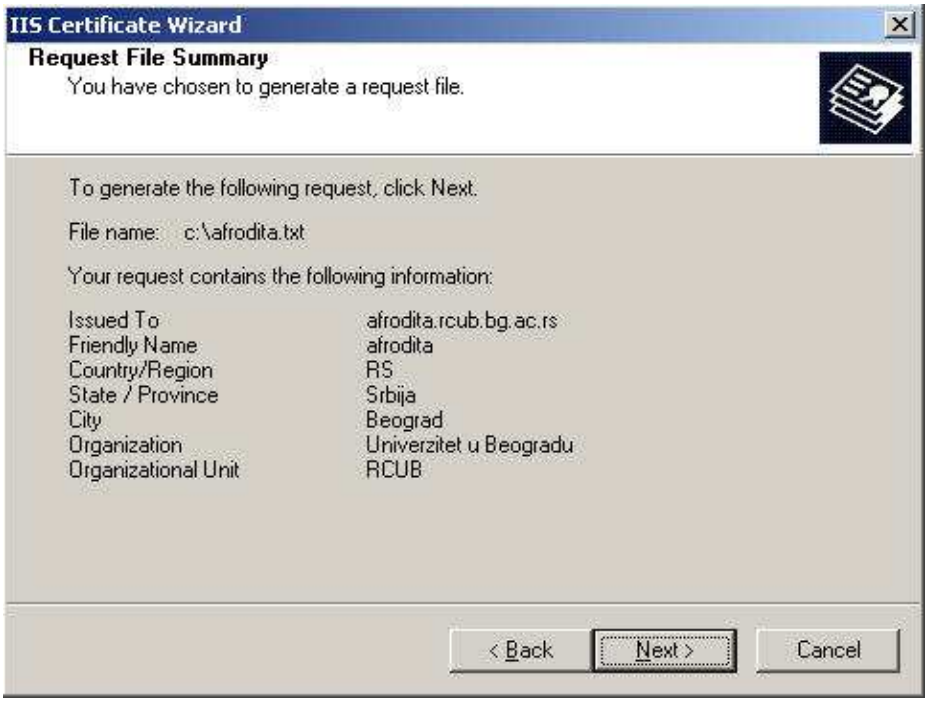

Figure 22

- Check the data you have entered. In the event of a mistake, click **Back** and make the appropriate changes. Following the successful verification of the data, click **Next**.

- The obtained Certificate Request is located in the *.txt* file between the lines "**-----BEGIN CERTIFICATE REQUEST-----**" and "**-----END CERTIFICATE REQUEST-----**", including these lines.

- Click **Next**.

Save the private key in the following manner:

- Go to: **Certificates snap-in** in the MMC (Microsoft Management Console)
- Select **Requests**
- Select **All tasks**
- Select **Export**

Note: The password that protects the private key, as well as the key itself, should be known only to the institution's administrative contact.

5.4 Submitting the Request

The certificate request is submitted on behalf of the institution by the institution's authorised person (administrative contact), designated in the process of registering the institution and submitting the application for using the TCS. 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-----", including these lines, should be copied from the file in which the Certificate Signing Request is located (*myserver.csr* on Linux servers or *file_name.txt* on Microsoft IIS servers) to the body of the message in plain text format. Files with the *.csr* extension and *.txt* extension can be opened with any text editor. 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:

- the full, official name of the institution under which it has been registered at the ac.rs domain registry portal (https://registar.ac.rs), and the address of the institution, i.e., street and street number, post code, 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.

6 Certificate Installation

The server SSL certificate, along with the file containing its *ca-chain*, is delivered to the orderer's e-mail in a *.zip* file. The name of the *.zip* file can be a seven-figure number (e.g., 9026687.zip) or the primary DNS name of the server, defined in the Certificate Signing Request (e.g., sf.bg.ac.rs.zip).

The certificate then needs to be installed on the server and the server should be configured in such a way that, besides its own SSL certificates, it also sends the *ca-chain* establishing the chain of trust for that certificate. The TERENA SSL CA certificate, used for signing each SSL certificate, is signed by UserTrust, the intermediate Certification Authority: the AddTrust External Root Certification Authority. The intermediate and root certificates are stored together in the *ca-chain* file. While most SSL clients have the most commonly used root certificates already preinstalled, the client also needs the information about the intermediate certificate signed by the CA root certificate so that each server SSL certificate can be verified.

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

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

If an institution has requested a TERENA 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 sections concerning individual services should be followed.

6.1 Web Server under Linux (Apache/mod_ssl)

The received *.zip* file should be unzipped, which will result in two files with the extensions *.crt* and *.ca-bundle*. The file with the *.crt* extension contains the Server Certificate, while the file with the *.ca-bundle* extension contains the chain of certificates for establishing a chain of trust for the issued Server Certificate. In the case of Linux distributions derived from Red Hat (CentOS, Fedora and Mandriva), certificates and keys are usually stored in the following locations, respectively:

```
unzip 9026687.zip

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

The *ssl.conf* configuration file should be edited and the appropriate paths to the Server Certificate, chain of certificates and the server key should be entered:

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

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

Finally, the web server should be restarted:

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

The following command can be used to see the installed certificate:

```
openssl x509 –in 9026687.crt –text
```

To install a Multi-Domain SSL Certificate, 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.

6.2 Java Web Server (e.g., Tomcat or JBoss)

The unzipped *.zip* file contains three files with certificates. The certificates need to be imported in the appropriate order: Root, Intermediate CA and server certificates (domain/site certificate).

The certificates are imported using the following *keytool* commands:

```
#import the root certificate
keytool -import -trustcacerts -alias root -file (ROOT CERTIFICATE FILE NAME) -
keystore domain.key

#import the intermediate certificate
keytool -import -trustcacerts -alias INTER -file (INTERMEDIATE CA FILE NAME) -
keystore domain.key

#import the server SSL certificate
#yyy is the alias specified during CSR creation
keytool -import -trustcacerts -alias yyy (where yyy is the alias specified during
CSR creation) -file domain.crt -keystore domain.key
```

It should be noted that the *alias* selected in the process of generating the CSR should be used during the last import, after which the *keytool* confirms the match with the following message: “Certificate reply was installed in keystore".

To install a Multi-Domain SSL Certificate, 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.

6.3 RADIUS Server

The following instructions describe 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 server certificate should be obtained in the manner described in Chapter 5: The AMRES TCS Certificate Service. The delivered *.zip* file that contains the certificate (e.g., 8866644.zip) should be transferred to the RADIUS server (e.g., using the WinSCP program) in the */etc/raddb/certs* directory, which usually stores certificates related to the FreeRadius server. This directory is also used for storing the private key of the server, generated in the process of requesting the given server certificate (assuming that the private key is stored in the *privatnikljuc.key* file). The following commands should be used:

```
#(move the zip file in the appropriate directory)
cp 8866644.zip /etc/raddb/certs/8866644.zip

#(move the private key)
cp privatnikljuc.key /etc/raddb/certs/privatnikljuc.key

#(unzip the zip file)
unzip 8866644.zip
```

Unzipping the *.zip* file produces two files with the extensions *.crt* and *.ca-bundle*. The *.crt* extension file contains the server certificate. The certificate from the *.crt* file should be converted into .pem format, using the following commands:

```
openssl x509 -in 8866644.crt -out 8866644.der -outform DER
openssl x509 -in 8866644.der -inform DER -out 8866644.pem -outform PEM
```

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

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

So that the authentication can function through the *ttls* protocol, first configure the *tls* in the *eap.conf* configuration file. Edit the *eap.conf* configuration file and enter the information on the certificates using the following commands:

```
certdir = /etc/raddb/certs
cadir = /etc/raddb/certs
private_key_file = /etc/raddb/certs/privatniključ.key
```

```
certificate_file = /etc/raddb/certs/8866644.pem
CA_file = /etc/raddb/certs/8866644.ca-bundle
```

The RADIUS process should now be restarted so that the changes in the configurations can take effect:

```
killall radiusd
radiusd
```

The client needs to install the Server Certificate in order to verify it successfully.

6.4 E-mail on a Linux Server

The procedure for installing a certificate for e-mail service on a Linux server is similar to the process of installing a certificate for a web server, described in Section 6.1.1: Web Server under Linux (Apache/mod_ssl), with the difference that additional configuration files are also 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 (that corresponds to the public key contained in the certificate) need to be entered in each configuration file of the service that is secured with the certificate.

The following commands are relevant for a Linux server on which a *postfix* package is installed to enable the exchange of e-mails based on the ***smtp*** protocol, and a *dovecot* package for downloading messages via *imap* and *pop3* protocols.

- For the ***postfix smtp*** server, the **main.cf** configuration file *(/etc/postfix/main.cf*) should be edited:

```
smtpd_use_tls = yes

#(path to the intermediate certificate)
smtpd_tls_CAfile = /etc/pki/tls/certs/8866644.ca-bundle

#(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/8866644.crt

#(path to the private key)
smtpd_tls_key_file = /etc/pki/tls/private/myserver.key
```

- For **imap and pop3** protocols on the server, the ***dovecot.conf*** configuration file *(/etc/dovecot.conf)* should be edited:

```
#(path to the certificate server)
ssl_cert_file= /etc/pki/tls/certs/8866644.crt

#(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/8866644.ca-bundle
```

To install a Multi-Domain SSL Certificate, the next service that is protected by the certificate should be opened and the data on the path to the directory in which the certificates are stored in the corresponding configuration file should be entered.

6.5 Microsoft IIS 5.x or 6.x

In this case, the name of the file containing the server certificate can be in the form of domain_name.cer. It contains the Root, Intermediate CA and server certificates.

The installation of the certificate is comprised of the following steps.

- Open the Control Panel and select **Administrative Tools**.
- Start the **Internet Services Manager,** which enables access to the wizard presented in Figure 13.
- Right-click the *website* option and select **Properties** by left-clicking.
- Select the **Directory Security** tab in the wizard shown in Figure 14.
- Click **Server Certificate**.

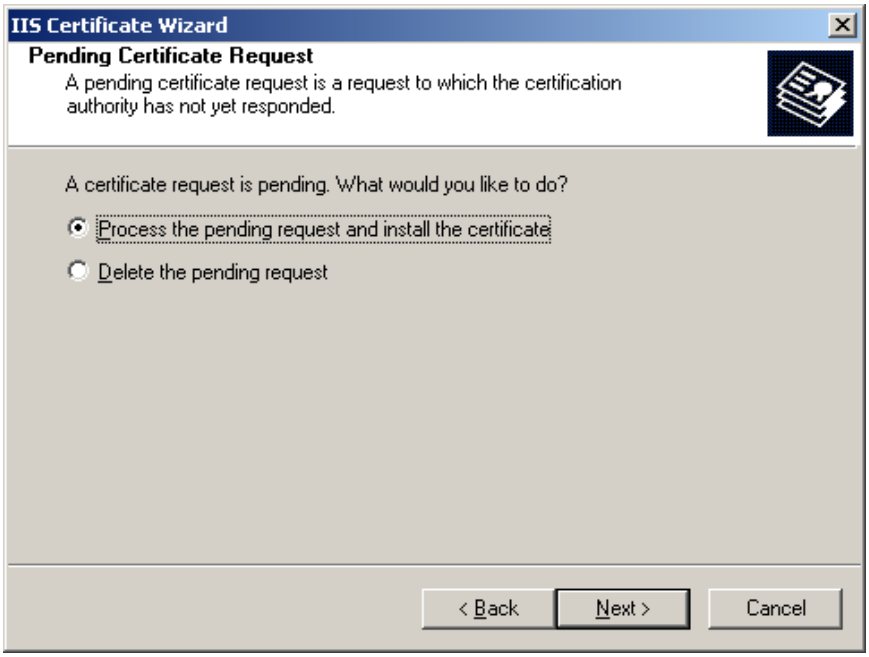

Figure 23

- Select **Process the Pending Request and Install the Certificate**. Click **Next**.
- Enter the path to the location of the server certificate. Click **Next**.
- Verify the data presented on the screen. Click **Next**.
- The **Confirmation screen** will appear. Click **Next**.
- Stop and start the appropriate website.

7 References

[1] William Stallings, “Network and internetwork security principles and practice”

[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”

[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

8 Glossary

CRL	Certificate Revocation List
FQDN	Fully Qualified Domain Name
IETF	Internet Engineering Task Force
IGTF	International Grid Trust Federation
PKI	Public Key Infrastructure
TERENA	Trans-European Research and Education Networking 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 certificate properties
RADIUS	Remote Authentication Dial In User Service

Appendix A Chain of TCS Certificates

The chain of CA certificates for the TCS server certificates is presented below. The first certificate in the sequence is the certificate of the Root CA – AddTrust External CA Root. This certificate is used for signing the next certificate in the chain, i.e., the certificate of the intermediate CA – UTN-USERFirst-Hardware. The certificate of the TERENA Certification Authority – TERENA SSL CA is at the end (signed by the intermediate certificate above).

The certificate of the Root Certification Authority is self-signed, so the values of the *Issuer* and *Subject* are the same. The *Issuer* field of the certificate, issued by the intermediate Certification Authority, contains the data of the Root Certification Authority, as the body that has issued the certificate. The TERENA CA certificate is issued by the intermediate Certification Authority, whose data is specified in the *Issuer* field.

AddTrust External CA Root (root certificate):

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=SE, O=AddTrust AB, OU=AddTrust External TTP Network, CN=AddTrust
External CA Root
        Validity
            Not Before: May 30 10:48:38 2000 GMT
            Not After : May 30 10:48:38 2020 GMT
        Subject:  C=SE,  O=AddTrust  AB,  OU=AddTrust  External  TTP  Network,
CN=AddTrust External CA Root
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:b7:f7:1a:33:e6:f2:00:04:2d:39:e0:4e:5b:ed:
                    1f:bc:6c:0f:cd:b5:fa:23:b6:ce:de:9b:11:33:97:
                    a4:29:4c:7d:93:9f:bd:4a:bc:93:ed:03:1a:e3:8f:
                    cf:e5:6d:50:5a:d6:97:29:94:5a:80:b0:49:7a:db:
                    2e:95:fd:b8:ca:bf:37:38:2d:1e:3e:91:41:ad:70:
                    56:c7:f0:4f:3f:e8:32:9e:74:ca:c8:90:54:e9:c6:
                    5f:0f:78:9d:9a:40:3c:0e:ac:61:aa:5e:14:8f:9e:
                    87:a1:6a:50:dc:d7:9a:4e:af:05:b3:a6:71:94:9c:
                    71:b3:50:60:0a:c7:13:9d:38:07:86:02:a8:e9:a8:
                    69:26:18:90:ab:4c:b0:4f:23:ab:3a:4f:84:d8:df:
```

```
                    ce:9f:e1:69:6f:bb:d7:42:d7:6b:44:e4:c7:ad:ee:
                    6d:41:5f:72:5a:71:08:37:b3:79:65:a4:59:a0:94:
                    37:f7:00:2f:0d:c2:92:72:da:d0:38:72:db:14:a8:
                    45:c4:5d:2a:7d:b7:b4:d6:c4:ee:ac:cd:13:44:b7:
                    c9:2b:dd:43:00:25:fa:61:b9:69:6a:58:23:11:b7:
                    a7:33:8f:56:75:59:f5:cd:29:d7:46:b7:0a:2b:65:
                    b6:d3:42:6f:15:b2:b8:7b:fb:ef:e9:5d:53:d5:34:
                    5a:27
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                AD:BD:98:7A:34:B4:26:F7:FA:C4:26:54:EF:03:BD:E0:24:CB:54:1A
            X509v3 Key Usage:
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Authority Key Identifier:
                keyid:AD:BD:98:7A:34:B4:26:F7:FA:C4:26:54:EF:03:BD:E0:24:CB:54:1A
                DirName:/C=SE/O=AddTrust      AB/OU=AddTrust      External      TTP
Network/CN=AddTrust External CA Root
                serial:01

    Signature Algorithm: sha1WithRSAEncryption
        b0:9b:e0:85:25:c2:d6:23:e2:0f:96:06:92:9d:41:98:9c:d9:
        84:79:81:d9:1e:5b:14:07:23:36:65:8f:b0:d8:77:bb:ac:41:
        6c:47:60:83:51:b0:f9:32:3d:e7:fc:f6:26:13:c7:80:16:a5:
        bf:5a:fc:87:cf:78:79:89:21:9a:e2:4c:07:0a:86:35:bc:f2:
        de:51:c4:d2:96:b7:dc:7e:4e:ee:70:fd:1c:39:eb:0c:02:51:
        14:2d:8e:bd:16:e0:c1:df:46:75:e7:24:ad:ec:f4:42:b4:85:
        93:70:10:67:ba:9d:06:35:4a:18:d3:2b:7a:cc:51:42:a1:7a:
        63:d1:e6:bb:a1:c5:2b:c2:36:be:13:0d:e6:bd:63:7e:79:7b:
        a7:09:0d:40:ab:6a:dd:8f:8a:c3:f6:f6:8c:1a:42:05:51:d4:
        45:f5:9f:a7:62:21:68:15:20:43:3c:99:e7:7c:bd:24:d8:a9:
        91:17:73:88:3f:56:1b:31:38:18:b4:71:0f:9a:cd:c8:0e:9e:
        8e:2e:1b:e1:8c:98:83:cb:1f:31:f1:44:4c:c6:04:73:49:76:
        60:0f:c7:f8:bd:17:80:6b:2e:e9:cc:4c:0e:5a:9a:79:0f:20:
        0a:2e:d5:9e:63:26:1e:55:92:94:d8:82:17:5a:7b:d0:bc:c7:
        8f:4e:86:04
```

UTN-USERFirst-Hardware (intermediate certificate)**:**

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            48:4b:ac:f1:aa:c7:d7:13:43:d1:a2:74:35:49:97:25
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=SE, O=AddTrust AB, OU=AddTrust External TTP Network, CN=AddTrust
External CA Root
        Validity
            Not Before: Jun  7 08:09:10 2005 GMT
            Not After : May 30 10:48:38 2020 GMT
        Subject:  C=US,  ST=UT,  L=Salt  Lake  City,  O=The  USERTRUST  Network,
OU=http://www.usertrust.com, CN=UTN-USERFirst-Hardware
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:b1:f7:c3:38:3f:b4:a8:7f:cf:39:82:51:67:d0:
```

```
                    6d:9f:d2:ff:58:f3:e7:9f:2b:ec:0d:89:54:99:b9:
                    38:99:16:f7:e0:21:79:48:c2:bb:61:74:12:96:1d:
                    3c:6a:72:d5:3c:10:67:3a:39:ed:2b:13:cd:66:eb:
                    95:09:33:a4:6c:97:b1:e8:c6:ec:c1:75:79:9c:46:
                    5e:8d:ab:d0:6a:fd:b9:2a:55:17:10:54:b3:19:f0:
                    9a:f6:f1:b1:5d:b6:a7:6d:fb:e0:71:17:6b:a2:88:
                    fb:00:df:fe:1a:31:77:0c:9a:01:7a:b1:32:e3:2b:
                    01:07:38:6e:c3:a5:5e:23:bc:45:9b:7b:50:c1:c9:
                    30:8f:db:e5:2b:7a:d3:5b:fb:33:40:1e:a0:d5:98:
                    17:bc:8b:87:c3:89:d3:5d:a0:8e:b2:aa:aa:f6:8e:
                    69:88:06:c5:fa:89:21:f3:08:9d:69:2e:09:33:9b:
                    29:0d:46:0f:8c:cc:49:34:b0:69:51:bd:f9:06:cd:
                    68:ad:66:4c:bc:3e:ac:61:bd:0a:88:0e:c8:df:3d:
                    ee:7c:04:4c:9d:0a:5e:6b:91:d6:ee:c7:ed:28:8d:
                    ab:4d:87:89:73:d0:6e:a4:d0:1e:16:8b:14:e1:76:
                    44:03:7f:63:ac:e4:cd:49:9c:c5:92:f4:ab:32:a1:
                    48:5b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Authority Key Identifier:
                keyid:AD:BD:98:7A:34:B4:26:F7:FA:C4:26:54:EF:03:BD:E0:24:CB:54:1A

            X509v3 Subject Key Identifier:
                A1:72:5F:26:1B:28:98:43:95:5D:07:37:D5:85:96:9D:4B:D2:C3:45
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 CRL Distribution Points:
                URI:http://crl.usertrust.com/AddTrustExternalCARoot.crl

    Signature Algorithm: sha1WithRSAEncryption
        3c:ec:7b:e0:ae:a3:0e:96:6d:30:d7:85:c6:d2:68:5b:45:5a:
        82:a6:34:0f:b0:c9:92:23:5e:11:6d:08:11:b2:74:09:23:3a:
        35:25:73:58:5e:ca:b9:7c:28:fa:47:ec:f9:a0:03:58:50:b6:
        53:ef:8c:db:39:e4:67:e9:d8:ca:28:46:d4:a7:e0:f5:38:75:
        f8:e7:cb:5c:bf:1d:11:3c:6a:40:9b:2d:44:56:d3:f7:ff:05:
        28:32:0c:15:c8:64:45:93:e8:21:24:8f:2d:da:7a:84:7b:4f:
        cf:cd:b2:25:7c:77:10:d3:94:d1:04:91:a8:25:1c:09:22:0f:
        7d:44:35:11:14:ef:af:00:fe:5e:ea:5f:8e:b0:d9:92:59:ba:
        fc:13:96:a0:18:01:56:ce:da:f6:28:0b:b1:af:dd:5c:4f:5c:
        b2:f3:8f:5a:71:cf:ed:18:ad:63:88:1d:8e:95:f7:ea:95:e7:
        1f:ad:90:b8:84:08:47:85:7f:22:2f:1a:1d:48:30:d6:4c:08:
        d8:37:19:67:32:2b:eb:5c:d0:b2:fc:6e:57:9f:04:35:5e:90:
        00:7e:11:c7:de:13:2a:cd:a4:6d:45:26:c7:88:56:a0:f0:6a:
        f7:d8:e7:fc:27:7e:67:08:d0:bd:fa:b6:c3:61:02:01:65:b9:
        b8:2f:cf:5a
```

TERENA SSL CA (TERENA CA Certificate)**:**

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            4b:c8:14:03:2f:07:fa:6a:a4:f0:da:29:df:61:79:ba
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, ST=UT, L=Salt Lake City, O=The USERTRUST Network,
OU=http://www.usertrust.com, CN=UTN-USERFirst-Hardware
        Validity
```

```
            Not Before: May 18 00:00:00 2009 GMT
            Not After : May 30 10:48:38 2020 GMT
        Subject: C=NL, O=TERENA, CN=TERENA SSL CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:c3:e3:48:c4:2f:5c:c1:cb:a9:99:fd:1b:a2:83:
                    5d:8a:3d:ad:3a:d0:e2:a4:43:1f:4d:0e:fe:35:25:
                    30:a5:69:1b:c4:e8:e5:c1:8f:54:7e:e1:6a:a2:9a:
                    5c:5c:de:3d:fc:02:ce:96:b8:5f:8f:83:5b:cc:60:
                    40:90:f8:e4:b6:3a:25:9c:5f:14:51:ec:b1:e7:af:
                    9e:50:a1:31:55:c7:02:bd:ac:52:8a:7f:35:8e:82:
                    fa:84:ad:15:fe:a2:7f:83:10:3a:55:53:94:2c:01:
                    16:74:94:54:63:28:a3:f2:5b:29:3d:94:88:80:20:
                    e2:14:59:21:19:b4:a4:98:e1:60:e6:f2:eb:a2:80:
                    83:43:e0:ad:68:f3:79:19:8b:68:43:51:3f:8a:9b:
                    41:85:0c:35:8c:5d:b5:f1:b6:e5:a7:c3:83:b5:6b:
                    23:6f:d4:a5:eb:50:e5:94:f1:4a:5f:ee:27:4b:14:
                    12:15:24:4c:0d:cf:62:8d:b7:00:21:ad:3a:32:0f:
                    58:0b:5f:1e:9b:d1:df:9d:8e:a9:19:35:50:2f:41:
                    a9:ad:3b:c6:e0:45:b2:53:39:7f:21:bf:22:1a:87:
                    5c:34:ae:52:6f:07:7d:a2:0b:4e:9f:2b:79:a6:7d:
                    13:dd:f5:7f:83:7c:2f:5a:5d:77:78:78:91:a0:14:
                    bf:7d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Authority Key Identifier:
                keyid:A1:72:5F:26:1B:28:98:43:95:5D:07:37:D5:85:96:9D:4B:D2:C3:45

            X509v3 Subject Key Identifier:
                0C:BD:93:68:0C:F3:DE:AB:A3:49:6B:2B:37:57:47:EA:90:E3:B9:ED
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Certificate Policies:
                Policy: 1.3.6.1.4.1.6449.1.2.2.29

            X509v3 CRL Distribution Points:
                URI:http://crl.usertrust.com/UTN-USERFirst-Hardware.crl

            Authority Information Access:
                CA Issuers - URI:http://crt.usertrust.com/UTNAddTrustServer_CA.crt
                OCSP - URI:http://ocsp.usertrust.com

    Signature Algorithm: sha1WithRSAEncryption
        4e:23:ee:48:9c:f6:85:8b:71:c4:0a:6e:73:93:72:c0:3a:8e:
        80:8a:d9:b3:ca:b2:d4:01:9c:28:cf:f2:5c:0e:21:44:93:0b:
        b6:1a:21:e3:98:01:94:0e:67:49:81:1e:be:3d:0d:4e:60:da:
        ef:a0:31:4e:95:ef:f3:dd:7a:5a:82:20:43:b6:a1:63:43:b3:
        50:69:43:62:4b:56:62:b0:34:8a:b9:13:43:59:93:ec:14:79:
        88:f3:48:93:e8:9d:c9:fa:87:72:0c:6b:56:a0:c3:15:8d:68:
        a5:87:1f:71:2d:e6:5a:6d:3c:69:71:40:04:55:dc:a0:43:94:
        20:45:38:78:d7:bd:8a:d8:39:c6:df:09:b7:5a:9a:a9:03:b8:
        28:10:78:cd:bf:01:1b:5a:11:3e:38:f4:d8:1b:34:79:cf:33:
        d2:01:fd:ac:98:cd:6d:47:11:90:4c:bb:b9:5b:d8:70:e7:d5:
        af:b6:cc:c4:86:e6:75:c0:9e:29:b6:2b:0f:2a:a5:69:02:0d:
        e3:e9:a2:b4:5d:c0:f3:ce:2c:6a:85:38:76:61:c6:49:82:ab:
        51:b3:82:a6:b9:41:98:28:98:fb:6b:fe:8a:16:ff:31:7e:54:
        47:a8:3c:dc:43:26:a9:9b:05:b7:9e:c0:34:43:91:30:d4:32:
```

```
        c3:11:5a:e1
```

Appendix B SSL Protocol

This content is available in Serbian only, in the online version of the document on the AMRES Wiki: https://www.bpd.amres.ac.rs/doku.php?id=amres_cbp_wiki:start.

Appendix C EAP-TTLS

This content is available in Serbian only, in the online version of the document on the AMRES Wiki: https://www.bpd.amres.ac.rs/doku.php?id=amres_cbp_wiki:start.