### Security Issues of the Digital Certificates within Public Key Infrastructures

Cristian TOMA

Economic Informatics Department, Academy of Economic Studies, Bucharest, Romania cristian.toma@ie.ase.ro

The paper presents the basic byte level interpretation of an X.509 v3 digital certificate according to ASN.1 DER/BER encoding. The reasons for byte level analysis are various and important. For instance, a research paper has mentioned how a PKI security may be violated by MD5 collision over information from the certificates. In order to develop further studies on the topic a serious knowledge about certificate structure is necessary. **Keywords:** digital certificates, certificates authority, ASN.1 DER/BER, PKI

**1** ASN.1 – Abstract Syntax Notation 1

ASN.1 is a sort artificial language used for describing data and data structure, instead of programs. The syntax is standardized in ISO/IEC 8824, and the coding rules are defined by ISO/IEC 8825. Both of these standards were developed from Recommendation X.409 of the CCITT. In order to get more information about ASN.1 it is recommended to consult [1]. In ASN.1, an octet is an eight-bit unsigned integer. Bit 8 of the octet is the most significant and bit 1 is the least significant.

The following meta-syntax and rules are used for in describing ASN.1 notation:

• **BIT** – mono-space denotes literal characters in the type and value notation; in examples, it generally denotes an octet value in hexadecimal

■ **n1** – bold italics denotes a variable

• [] – bold square brackets indicate that a

term is optional

• {} - bold braces group related terms

• | – bold vertical bar delimits alternatives with a group

• ... – bold ellipsis indicates repeated occurrences

• = - bold equals sign expresses terms as sub-terms.

• Layout is not significant; multiple spaces and line breaks can be considered as a single space

• Identifiers (names of values and fields) and type references (names of types) consist of upper- and lower-case letters, digits, hyphens, and spaces; identifiers begin with lower-case letters; type references begin with upper-case letters. Comments are delimited by pairs of hyphens (--), or a pair of hyphens and a line break.

Some of the **data types** used in ASN.1 are presented in the following listing:

Data Type	Sort	Tag Number	Meaning
BOOLEAN	Primitive	0x01	Boolean value: yes/no
INTEGER	Primitive	0x02	Negative and positive integers
BIT STRING	Primitive	0x03	Bit sequence
OCTET STRING	Primitive	0x04	Byte sequence (1 byte=8 bits=1octet)
NULL	Primitive	0x05	A null value
OBJECT IDEN-	Primitive	0x06	An object identifier, which is a sequence of integer com-
TIFIER			ponents that identify an object such as an algorithm or
			attribute type
PrintableString	Primitive	0x13	An arbitrary string of printable characters
T61String	Primitive	0x14	An arbitrary string of T.61 (eight-bit) characters
IA5String	Primitive	0x16	An arbitrary string of IA5 (ASCII) characters
UTCTime	Primitive	0x17	A "coordinated universal time" or Greenwich Mean Time
			(GMT) value

Data Type	Sort	Tag Num-	Meaning
		ber	
SEQUENCE	Constructed	0x30	An ordered collection of one or more types
SEQUENCE OF	Constructed	0x10	An ordered collection of zero or more occurrences of a giv-
			en type
SET	Constructed	0x31	An unordered collection of one or more types
SET OF	Constructed	0x11	An unordered collection of zero or more occurrences of a
			given type
A0, A1,	Constructed	0xAz	Where $z = 0$ F in hex and it represents the z-th element in
			SEQUENCE data type

Some of structured types defined in ASN.1 are presented in the following listing:

The idea in ASN.1 is to prefix each data objects. ject with a unique label and information about its length. Users are allowed to define a controller using ASN.1 is:

11	is tengul. Osets are allow	
	SC_Controller ::= SEQUEN	CE { //Describe a new data type for a controller
	Name IA5String,	//The name of the microchip is an ASCII string
	CPUType CPUPower,	//CPUType refers to the definition of CPUPower
	NPU BOOLEAN,	//yes or no if coprocessor is present
	EEPROMSize INTEGER,	//the size in bytes of EEPROM
	RAMSize INTEGER,	//the size in bytes of RAM used in smart card
	ROMSize INTEGER	//the size in bytes of ROM used in smart card
	}	
	CPUPower ::= ENUMERAT	ED { //Definition of CPUPower as an enumeration
	8Bit (8),	//Possible selection values for the CPUType
	16Bit (16),	
	32Bit (32)	
	}	

An example for a particular XSS microcontroller using previous definition in ASN.1 is described in the following mode:

$\mathcal{U}$	
XSS SC_Controller ::= {	//Specific instance-controller of SC_Controller
Name "XS 8 Bit",	//The name of controller is XS 8 Bit
CPUType 8,	//this is an 8-bit CPU
NPU true,	//NPU present
EEPROMSize 1024,	//The size of EEPROM size is 1024 bytes
RAMSize 256,	//The size of RAM size is 256 bytes
ROMSize 8192	//The size of ROM size is 8192 bytes
1	•

The **BER** – **Basic Encoding Rules** for ASN.1 are defined in the ISO/IEC 8825 standard. A BER coded data object should have:

• A length field L

• The data content **V** - **value** 

• A label called **T** – **tag** 

Figure 1 presents the BER-based TLV coding principle accordingly to ASN.1:

T - tag	L - length	V - value	ļ	TLV Object
1-2 bytes	1-3 bytes	n bytes		IL Colject

Fig. 1. ASN.1 BER-based TLV coding

A sample data for a particular microcontroller model is coded using the **ASN.1 BER**: 0x30 0x1C 0x16 0x08 0x58 0x53 0x20 0x38 0x20 0x42 0x69 0x74 0x0A 0x01 0x08 0x01 0x01 0xFF 0x02 0x02 0x04 0x00 0x02 0x02 0x01 0x00 0x02 0x02 0x20 0x00.

'30 1C'	//Tag 0x30 for a array with a length of 28
	//bytes (0x1C)
'16 08 58 53	20 38 20 42 69 74' //Tag 0x16 for an IA5String with a length of
	//8 bytes (0x08) with a content of "XS 8 Bit"
'0A 01 08'	//Tag 0x0A for an enumerated data type with
	//a length of 1 byte (0x01) and a content of 8
'01 01 FF'	//Tag 0x01 for Boolean data type with a length of 1 byte (0x01)
	//and a content of 0xFF (true)
'02 02 04 00	)' //Tag 0x02 for an integer data type with a length of 2 bytes
	//(0x02) and of 1024 bytes (0x0400)
'02 02 01 00	" //Tag 0x02 for an integer data type with a length of 2 bytes
	//(0x02) and of 256 bytes (0x0100)
'02 02 20 00	" //Tag 0x02 for an integer data type with a length of 2 bytes
	//(0x02) and of 8192 bytes (0x0200)

When the author discuss about the start of a sequence for an X509 v3 certificates, it would be  $0x30 \ 0x82 \ 0x01 \ 0xC3 - SE-QUENCE {...} - this means the certificate is a SEQUENCE structure (first byte with 0x30 value) with 451 bytes. The length is specified in 3 bytes 0x82 0x01 0xC3. The last hex digit (nibble = half-byte) from the first byte of the length field specifies the length in bytes of the structure => 2 bytes. The next 2 bytes actually express the length of the structure SE-QUENCE, and they have the value 0x01C3 => 451 bytes.$ 

The BER special encoding applies to the OID-s – Object Identifiers - in Internet.

In the figure 2 there is some of the ISO tree hierarchy for constructing objects in Internet structure. It is useful to imagine how would be written the OID with value 1.2.840.113549.1.1.5  $(\{iso(1)\}$ memberbody(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) sha1-with-rsa-signature(5)}) for signature obtained from SHA-1 digest on the message and RSA algorithm applied to the digest. This encoding will be at byte level: 0x06 0x09 0x2A 0x86 0x48 0x86 0xF7 0x0D 0x01 0x01 0x05. The first byte shows that here there is an OID - OBJECT IDENTIFI-ER field. The OID has 9 bytes length because of second byte in array. Because the first bit in length field is not set the length field has only one byte. The length field has value 9 which means the OID structure has the payload data in 9 bytes.

According to BER, the first two numbers of any OID (x,y) are encoded as one value using

the formula (40\*x)+y. The first two numbers in an OID are here 1.2. Therefore, the first two numbers of an OID are encoded as 42 or 0x2A, because (40\*1)+2 = 42. After the first two numbers are encoded, the subsequent numbers in the OID are each encoded as a byte. However, a special rule is required for large numbers because one byte (eight bits) can only represent a number from 0-255. This is the case for 840 and 113549. For 840 from the OID, the first bit of the first byte should be set. The number occupies enough number of bytes till the last byte of representation is not having the first bit set. In 840 case 0x48 has the first bit NOT set. So, the formula is the last hex digit from the first byte multiplied with  $2^{7} = 128$  (1 bit is for multiple bytes representation) and added with the value from the second byte as long as the second byte has a value less than 0x80. In 840 case, the formula is: 0x06\*128 + 0x48(form 0x86 0x48) = 768 + 72 = 840. In case of 113549 we choose the bytes 0x86 0xF7 0x0D because 0x0D is the last byte with first bit (sign bit) not set. The formula for 113549 is:

 $0x06*2^{14} + 0x77*2^{7} + 0x0d*2^{0} = 6*16384 + 119*128 + 13*1 = 98304 + 15232 + 13 = 113549.$ 

For the remaining encoding  $\{\dots$  pkcs(1) pkcs-1(1) sha1-with-rsa-signature(5) $\}$ , we will have only one byte for each number: 0x01 0x01 0x05.

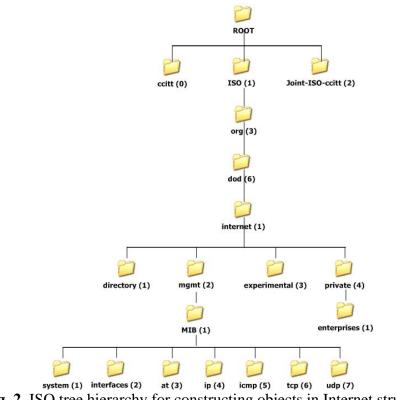


Fig. 2. ISO tree hierarchy for constructing objects in Internet structure

For checking out an OID, please visit [3]. The **DER** – **Distinguished Encoding Rules** form a subset of the BER and give exactly one way to represent any ASN.1 value as an octet string. DER is intended for applications in which a unique octet string encoding is needed, as is the case when a digital signa-

ture is computed on an ASN.1 value. A basic summary of the BER and DER are found in Burton Kaliski [1].

The two most significant bits of the bytes T - tag encode the class of the data object. The bits positions are presented in figure 3:

	T - tag	L - length			
b7 b6 b5 b4 b3 b2 b1 b0	Meaning	b7 b6 b5 b4 b3 b2 b1 b0	Byte 1 Byte 2 Byte 3	Meaning	
0 0 0 1 1 0 1 1 0 1 Y Y Y Y Y 1 1 1 1 1	Application class Context-specific class Private class Primitive data object Constructed data obj.	0 The value of tag code in range 31-127	0 - 127 0x81 128-255 0x82 256-65535	One byte is needed Two bytes are needed Three bytes are needed	
Byte	1	Byte 2	Byte	3,4,5	

Fig. 3. Structure of T and L fields in ASN.1 [5]

In figure 3, the *universal class* indicates general data objects such as integers and character strings. The *application class* indicates that the data object belongs to a particular (e.g. electronic wallet from a smart card) or standard (e.g. ISO/IEC 7816-6) application.

The classes context-specific and private are not matter of standard applications. The bit following the first two bits indicates if the tag is for a primitive or constructed data object. The five least significant bits from byte 1 are the actual label, the tag value. If the tag value is greater then 30, then all the bits from first byte have value 1 and the bits b0-b6 from second byte store a value between 31-127 ranges. The byte 3, 4 and 5 are used for marking the data length. If data has a byte length less than 127 then only the byte 3 (first byte from L) is used to store the actual length. If the length value is between128255, then 2 bytes are used for storing the length and if the length is between 256 and 65535, then all three bytes from L field are used for storing the length value.

Figure 4 presents a sample about how the first name of a person can be stored in a smart-card.

value 15	UCLWCCII	120-						
Tag	Length	Value						
0x85	0x05	0x46	0x72	0x61	0x6E	0x6B		
Tag for first Length of the First name "Frank" in ASCII								
names first name								
- 4 TI V	I = - 1	<b>f</b>	41 f:		···· · · · · · · · · · · · · · · · · ·	1_?? [		

Fig. 4. TLV Encoding for the first name "Frank" [5]

Subsequent extensions to data structures can be undertaken very easily with ASN.1 since all that is necessary is to insert additional TLV-coded data objects into existing data structure. Full compatibility with the previous versions is retained as long as the previous TLV objects are not deleted. The same is true of new version of data structures in which changes have been made accordingly to the previous coding. This kind of encoding is used on the large scale in smart card industry. Figure 5 presents a basic scheme for forming constructed TLV-coded data structures from several primitive TLV-coded data objects.

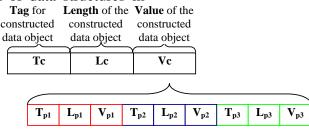


Fig. 5. Nested TLV-coded data objects [5]

The indices 'C' and 'P' stand for 'constructed' and 'primitive' in figure 5. This kind of coding has the advantage of no matter order have the TLV-coding structures the programs could be able to interpret them correct. When evaluating the TLV-coding data structure, the program compares the first tag with all tags known to it. If it finds a match, then it recognizes the first object from byte array, and it is able to "say" how long it is.

# **2. Digital Certificates and Certification** Authorities

The infrastructures based on the cryptography with asymmetric (public) keys are essential for the viability of the message transactions and communications especially in networks and generally on the Internet. The PKI – Public Key Infrastructure consists of the multitude of services required to be ensured when the technologies of encryption with public keys are used on a large scale. These services are of technological nature, as well as legal, and their existence is necessary in order to permit the exploitation of public keys technologies at their full capacity.

The main elements of the public keys infrastructures are:

- Digital certificates
- Certification Authorities (CA)

 Management facilities (protocols) for certificates

The public keys infrastructures must ensure support for encryption functions, as well as for those for digital signature.

In the functioning of public keys systems is necessary a system for generating, circulating and certifying the users' keys. Given a user C who intents to pose as A, and wants to fake sign as. Faker C may easily do that, generating his own pair of keys and placing the public one in the public folder, instead of the real one of A (instead of the public key of A).

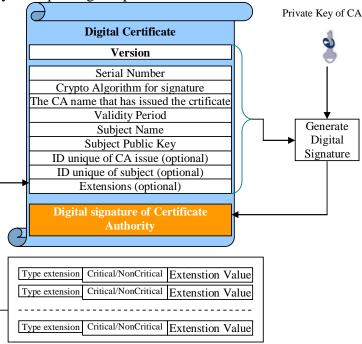


Fig. 6. X509 v3 Certificate

The documents and messages signed by C with his secret key will be verified with the public key that seems to be of A and any person will be deceived the authenticity of the messages signed on behalf of A. The main problem is thus of the total trust in public keys, those used for certifying the digital signatures. These (signatures) must be available on the network, so that any client may get the public key of a sender of a signed message. So, the technical solution exists: creating an international infrastructure, based on Certification Authorities - CA, that may allow the easy access and in a secure mode to the public keys of the entities that wish to communicate in the network or Internet. These authorities will distribute, by request, certificates of authentic keys. The most largely known and used format for digital certificates of public keys is that defined in the standard X.509 by ISO/IEC/ITU. The format X.509 for certificates has evolved in three versions. Version 3 (adopted in 1996) is the best known. Figure 6 illustrates the format of the certificate X.509 v3.

While the previous versions ensured support only for the name system X.500, X.509 v3 accepts a large variety of forms for names, including e-mail addresses and URLs.

A system based on public keys certificates presumes the existence of a Certification Authority, which issues certificates for a certain group of owners of keys pairs (public and private). Each certificate includes the value of the public key and information that uniquely identifies. The certificate's subject (who is a private person or a company, an application, a device or another entity that holds the secret key correspondent to the public key included in the certificate). The certificate represents a liaison impossible to falsify, between a public key and a certain attribute of its owner. The certificate is digitally signed by a Certification Authority (certified by the government), that so confirms the subject's identity. Once the certificates set established, an user of that public key infrastructure (PKI) may obtain the public key for any user certified by that Certification Authority, simply getting the certificate for that user extracting from it the desired public The ASN.1 representation of an key. X509.v3 certificate is the following:

Certificate ::= SIGNED { SEQUENCE { version [0] Version DEFAULT v1, serialNumber CertificateSerialNumber, signature AlgorithmIdentifier, issuer Name, validity Validity, subject Name. subjectPublicKeyInfo SubjectPublicKeyInfo, issuerUniqueIdentifier [1] IMPLICIT UniqueIdentifier OPTIONAL, -- if present, version shall be v2 or v3 subjectUniqueIdentifier [2] IMPLICIT UniqueIdentifier OPTIONAL, -- if present, version shall be v2 or v3 extensions [3] Extensions OPTIONAL -- If present, version shall be v3 -- } } Version ::= INTEGER { v1(0), v2(1), v3(2) } CertificateSerialNumber ::= INTEGER AlgorithmIdentifier ::= SEQUENCE { algorithm ALGORITHM.&id ({SupportedAlgorithms}), parameters ALGORITHM.&Type ({SupportedAlgorithms}{ @algorithm}) OPTIONAL } -- Definition of the following information object set is deferred, perhaps to standardized -- profiles or to protocol implementation conformance statements. The set is required to -- specify a table constraint on the parameters component of AlgorithmIdentifier. -- SupportedAlgorithms ALGORITHM ::= { ... } Validity ::= SEQUENCE { notBefore Time, notAfter Time } SubjectPublicKeyInfo ::= SEQUENCE { algorithm AlgorithmIdentifier, subjectPublicKey BIT STRING } Time ::= CHOICE { utcTime UTCTime, generalizedTime GeneralizedTime } **Extensions ::= SEQUENCE OF Extension** Extension ::= SEQUENCE { extnId EXTENSION.&id ({ExtensionSet}), critical BOOLEAN DEFAULT FALSE, extnValue OCTET STRING -- contains a DER encoding of a value of type &ExtnType -- for the extension object identified by extnId -- } ExtensionSet EXTENSION ::= { ... }

In figure 7 is represented an X 509 v3 certificate in hex editor and in Windows Crypto Shell program.

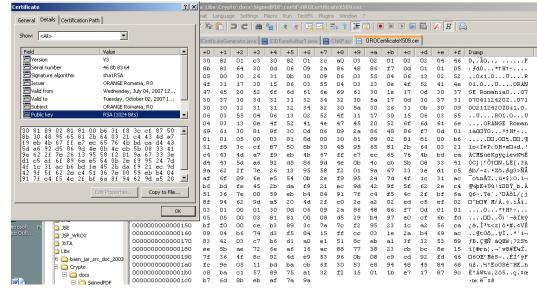


Fig. 7. X509 v3 Certificate in Hex and in MS Crypto Shell

Tag	Length	Value	ASN 1 Notation	Comments
30	82 01 C3		SEQUENCE {	Byte with value 0x30 is ASN 1 header, complex
				sequence, the the last digit from the first byte of
				the length bytes specifies the length in bytes of
				the structure $\Rightarrow$ 2 bytes with 01C3 value $\Rightarrow$ 451
				bytes
30	82 01 2C		SEQUENCE {	ASN Complex Sequence Header with 300 bytes
				length, to be signed parts begin here
A0	03		[0] {	ASN 1 DER Encoding for first element from the
				SEQUENCE, has 3 bytes length. Tag 0x0A for
				an enumerated data type with a length of 3 bytes
				(0x03) and a content $0x02 0x01 0x02$
02	01	02	INTEGER 2	Attribute version, 1 byte length with value $2 \Rightarrow$
	• -		}	X 509 version 3 (starting from 0)
02	04	46 8B 83 64	INTEGER 1183548260	Attribute Serial Number with the value
02	04	40 00 05 04	HTESER 1103540200	1183548260
30	0D		SEQUENCE {	Another sequence within first two sequences al-
30	0D		SEQUENCE {	ready opened and has 13 bytes length
06	00		OD JECT IDENTIFIED	
06	09	2a 86 48 86 f7 0d 01	OBJECT IDENTIFIER	Signature algorithm identifier - sha1RSA = sha-
		01 05	rsaWithSha1	1WithRSAEncryption OBJECT IDENTIFIER
			(1 2 840 113549 1 1 5)	$::= \{iso(1) member-body(2) us(840) rsad-$
				si(113549) pkcs(1) pkcs-1(1) sha1-with-rsa-
				signature(5)}
				For 840 from the OID the formula is
				$0x06*128 + 0x48$ (form $0x86\ 0x48$ )= 768 + 72
				= 840
				$0x06*2^{14} + 0x77*2^{7} + 0x0d*2^{0} = 6*16384$
				+ 119*128 + 13*1 = 98304 + 15232 + 13 =
				113549
05	00		NULL	Element NULL in ASN.1 BER with 0 bytes
			}	length. So, NULL 2 bytes and the SEQUENCE
			-	is ending.
30	26		SEQUENCE {	Another sequence within first three sequences
	-			already opened and has 38 bytes length $(0x26)$
31	0B		SET {	A SET with 12 bytes length
30	09		SEQUENCE {	
06	03	55 04 06	OBJECT IDENTIFIER	OID countryName {joint-iso-itu-t(2) ds(5) attri-
00	05	55 04 00	countryName (2 5 4 6)	buteType(4) countryName(6)}
13	02	52 4F	PrintableString 'RO'	countryName = 'RO' Printable string with 2
15	02	52 4F	FilinableSunig KO	bytes length for issuer
		-	)	
			}	Closed the SET and the SEQUENCE
21	17		} SET (	A SET with 22 bytes log -th
31	17		SET {	A SET with 23 bytes length
30	15		SEQUENCE {	Another sequence within first three sequences
0.5			0.0.000	already opened and has 21 bytes length (0x15)
06	03	55 04 03	OBJECT IDENTIFIER	OID commonName {joint-iso-itu-t(2) ds(5) at-
			commonName (2 5 4 3)	tributeType(4) commonName(3)}
13	0E	4F 52 41 4E 47 45 20	PrintableString	commonName = 'ORANGE Romania' for issuer
		52 6f 6d 61 6e 69 61	'ORANGE Romania'	Printable String with 14 bytes length
			}	Closed the SET and the SEQUENCE and the
			}	first above SEQUENCE – the one with 38 bytes
			}	length (0x26) for issuer
30	1E		SEQUENCE {	Another sequence within first sequence already
-				opened and has 30 bytes length $(0x15)$
17	0D	30 37 30 37 30 34 31	UTCTime	A "coordinated universal time" or Greenwich
1,		31 32 34 32 30 5a	'070704112420Z'	Mean Time (GMT) value, for <i>valid from</i>
		51 52 57 52 50 5a	0.0.0112-1202	YYMMDDHHmmSS format, meaning 04 July
				2007 11:24:20
17	0D	20 27 21 20 20 20 21		
17	0D	30 37 31 30 30 32 31	UTCTime	A "coordinated universal time" or Greenwich
		31 32 34 32 30 5a	'071002112420Z'	Mean Time (GMT) value, for <i>valid to</i>
	1	1		YYMMDDHHmmSS format, meaning 02 Oc-
				tober 2007 11:24:20 Closed the SEQUENCE with 0x1E length

## BER is in the following listing [4]:

30         26         SEQUENCE           Another sequence within first sequence already opened and has 38 bytes length (0x26).           31         08         SET [         A SET [					
31         00B         SET { A SET with 11 bytes length           30         07         SEQUENCE [ Another sequence within first two sequences length           06         03         55 04 06         OBLECT IDENTIFIER contryName [- \$KO Printable string with 2 butsyles (contryName [- \$KO Printable string with 2 bytes length for subject           13         02         52 4F         PrintableString RO' contryName [- \$KO Printable string with 2 bytes length for subject           30         17         SET [ A SET [ A SET [ A SET with 23 bytes length for subject           30         15         SEQUENCE [ Another sequences with in first two sequences with in first two sequences with in first two sequences with in bytes length (0x15)           30         0E         4F 52 41 4E 47 45 20         OBLECT IDENTIFIER control (100 control (0x6))           31         0E         4F 52 41 4E 47 45 20         FinableString CORNER(Somana)           31         0E         4F 52 41 4E 47 45 20         Control (0x6) for subject [ Another sequences (0x6) at the first above SEQUENCE [ Another sequence with a stapped (0x15)]           30         0E         2E 66 61 61 66 69 61         OBLECT IDENTIFIER [ another SEQUENCE [ Another sequence with a bytes length (0x6)]           30         0D         SEQUENCE [ Another sequence with in first sequence already opened and has 13 bytes length (0x6)]           30         0D         SEQUENCE [ Another sequence with anot two seque	30	26		SEQUENCE {	Another sequence within first sequence already $(0x^{26})$
30         09         SEQUENCE {         Another sequence within first two sequences and one set already openel and has 09 bytes length           06         03         55 04 06         OBJECT IDENTIFIER OID countryName {i0int-iso-itu+(2) ds(3) attributes (25 4 6)           13         02         52 4F         PrimableString RO'         countryName = RO' Primable string with 2 bytes length for subject           13         02         52 4F         PrimableString RO'         countryName = RO' Primable string with 2 bytes length           30         15         SEQUENCE {         Another sequence within first two sequences already openel and has 21 bytes length (0x15)           14         0E         44 52 41 42 47 45 20         SEQUENCE {         another sequence within first two sequences already openel and has 21 bytes length (0x15)           13         0E         54 66 61 66 96 61         1         commonName 2 - ORANGE Romania'         commonName (30 + 16 + 16 + 16 + 16 + 16 + 16 + 16 + 1	31	0B		SET {	
of         of <thof< th="">         of         of         of<!--</td--><td></td><td></td><td></td><td>,</td><td></td></thof<>				,	
Image: constraint of the sequence of the sequ					
$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$					
13         02         52 4F         PrinubleString 'RO'         countryName         RO'         Prinuble string with 2 bytes length for subject           31         17         SET {         A SET with 2 bytes length         Another sequence within first three sequences and sequence within first three sequences and sequence within first three sequences and sequences within first sequences al- ready opened and has 19 bytes length (0x26) for subject           30         81 9F         Closed the SET and the SEQUENCE is end with 38 bytes length (0x26) for subject         Another sequences within first sequences al- ready opened and has 19 bytes length (0x9F)           30         0D         SEQUENCE {         Another sequences within first sequences al- ready opened and has 19 bytes length (0x9F)           30         0D         SEQUENCE {         Another sequences within first sequences al- ready opened and has 19 bytes length (0x9F)           30         0D         SEQUENCE {         Another sequences within first two sequences al- ready opened and has 19 bytes length (0x9F)           30         0D         2a 86 48 86 F7 0D 01         OBECT TIDENTIFIER readsorption (1 2 840 113549 1 1 1)         OBECT TIDENTIFIER readsorption (1 2 840 113549 1 1 1)         Closed of 10 cor (0 x00), the thind and fourth byte are the length of R5A public exponent. In t	06	03	55 04 06	OBJECT IDENTIFIER	
Image: constraint of the stress of the st					
31IndEnd of SEQUENCE and SET3117SET {A SET with 23 bytes length3015SEQUENCE {Another sequence within first three sequences3015SEQUENCE {Another sequence within first three sequences31064F 52 41 4E 47 45 20OBJECT IDENTIFIEROD commonName {} jont-iso-iu-ic/2 ds(5) at-triable String with 12 bytes length3081 9FORANGE Romania'CommoName {} ORANGE Romania'CommoName {} ORANGE Romania', for sub-ic-triable String with 14 bytes length3081 9FSEQUENCE {Another sequence within first sequence already opened and has 159 bytes length (0x9F)300DSEQUENCE {Another sequence within first sequence already opened and has 159 bytes length (0x9F)3001SEQUENCE {Another sequence within first sequence already opened and has 159 bytes length31002a 86 48 86 F7 0 D1OBJECT IDENTIFIER ratercryption (1 2 840 113549 1 1 1)3101NULL (1 2 840 113549 1 1 1)320381 8D00 30 81 89 02 81 81 00 63 11 83 cc f 87 50 80 30 48 95 65 81 20 80 30 48 95 68 18 20 86 30 21 c4 43 40 a 7 9 e4 46 of ff e7 ec 65 76 4b bd ca d4 43 ch 79 e4 46 of ff e7 ec 65 76 4b bd ca 44 43 ch 79 eb 46 of ff e7 ec 65 76 4b bd ca 44 43 ch 79 eb 46 of ff e7 ec 65 76 4b bd ca 44 43 ch 79 eb 46 of ff e7 ec 65 76 4b bd ca 44 43 ch 79 eb 46 of ff e7 ec 65 76 4b bd ca 44 43 ch 79 eb 46 of ff e7 ec 65 76 4b bd ca 44 43 ch 79 eb 46 of ff e7 ec 65 76 4b bd ca 44 43 ch 79 eb 46 of ff e7 ec 65 76 4b bd ca 44 43 ch 79 eb 46 of ff e7 ec 65 76 4b bd ca 44 43 <b< td=""><td>13</td><td>02</td><td>52 4F</td><td>PrintableString 'RO'</td><td></td></b<>	13	02	52 4F	PrintableString 'RO'	
i         i         A SET with 23 bytes length           30         15         SEQUENCE {         Another sequence within first three sequences already opened and has 21 bytes length (No.15)           06         03         55 04 03         OBJECT IDENTIFIER commonName (Joint-iso-ita-it2) ds(5) at induct/ye(4) commoName (Porta-iso-ita-it2) ds(5) at induct/ye(4) commoName (PortANCE Romanial, for subject Prinable String with 14 bytes length           13         0E         4F 52 41 4E 47 45 20         ORANGE Romania         OCased the SET and the SEQUENCE and the first above SEQUENCE in the one with 38 bytes length (0x26) for subject           30         81 9F         SEQUENCE {         Another sequence within first woo sequences already opened and has 15 bytes length (0x26) for subject           30         0D         SEQUENCE {         Another sequence within first woo sequences already opened and has 15 bytes length (0x26) for subject           30         0D         SEQUENCE {         Another sequence within first woo sequences already opened and has 15 bytes length (0x26) for subject           30         0D         SEQUENCE {         Another sequence within first woo sequences already opened and has 15 bytes length for RSA           30         0D         NULL         BIT STRING 0 unused         BIT STRING vin 141 bytes length for RSA           31         00 do 51 R3 cc f8 7         50 R3 04 89 56 58 1         56 for 4b bd c1 d4 43 54           30         <					
30       15       SEQUENCE [       Another sequence within first three sequences of the sequence of the sequences of the sequence sequence within first three sequences (0.15)         06       03       55 04 03       OBJECT IDENTIFIER commonName (2.5 4 3)       OID commonName (2.12) ds(5) at tributeType(4) commonName (2.12) ds(6) ds(6) ds(7)				}	End of SEQUENCE and SET
30       15       SEQUENCE [       Another sequence within first three sequences of the sequence of the sequences of the sequence sequence within first three sequences (0.15)         06       03       55 04 03       OBJECT IDENTIFIER commonName (2.5 4 3)       OID commonName (2.12) ds(5) at tributeType(4) commonName (2.12) ds(6) ds(6) ds(7)	31	17		} SET (	A SET with 23 bytes length
$dec_{2}$ <td></td> <td></td> <td></td> <td>e e e e e e e e e e e e e e e e e e e</td> <td></td>				e e e e e e e e e e e e e e e e e e e	
06     03     55 04 03     OBJECT IDENTIFIER commonName (2 5 4 3)     OID commonName (3) (3)       13     0E     4F 52 41 4E 47 45 20 52 61 6d 61 6e 69 61     PrintableString ORANGE Romania'     commonName (2 5 4 3)       13     0E     4F 52 41 4E 47 45 20 52 61 6d 61 6e 69 61     PrintableString ORANGE Romania'     commonName (2 5 4 3)       30     81 9F     1     Closed the SET and the SEQUENCE 1 a	50	15		SEQUENCE (	
13 $commonName (2 5 4)$ tribue Type(4) commonName(3)13 $0E$ $4F 52 4I 44 27 45 20$ PrintableStringcommonName = 'ORANGE Romania', for sub- ject Printable String with 14 bytes length13 $0E$ $52 6I 6d 6I 6e 69 6I$ $PintableString$ commonName = 'ORANGE Romania', for sub- ject Printable String with 14 bytes length30 $81 9F$ $PintableString$ Closed the SET and the SEQUENCE - the one with 38 bytes length (0x26) for subject30 $81 9F$ SEQUENCE { another sequence within first sequence already opened and has 159 bytes length30 $0D$ SEQUENCE { ready opened and has 13 bytes length30 $0D$ SEQUENCE { ready opened and has 13 bytes length31 $00$ $0BIECT IDENTIFIER$ ready opened and has 13 bytes length32 $00$ $0BIECT IDENTIFIER$ ready opened and has 13 bytes length33 $00$ $010$ $0BIECT IDENTIFIER$ ready opened and has 13 bytes length34 $00$ $03 81 89 02 81 81$ $00 30 81 89 02 81 81$ $00 56 31 18 3 c cf 8750 80 30 48 95 65 8120 5 40 42 75 60 32 1c 43 443400 40 32 1c 44 3417 9 eb 46 67 ff c7 ec a65 76 4 b b c ad 44 37352d 69 2 45 86 9d 4e0b 4c cb 5b 86 33 419a 62 2 17 e 26 13 95360Daf 62 21 ec 5041 c 5 2c 16 76 26 18 9537af 67 80 6e c554 8 12 01 9a 67 33 3e41 c 5 af 67 80 6e c536af 9 42 9 45 20 4421 c 62 4a 52 04 4421 c 62 2a 20 2e d c 5ef00 10 10 10 10 10 10 10 10 10 10 10 10 1$	06	03	55 04 03	OBJECT IDENTIFIER	
S2 6f 6d 61 6e 69 61         ORANGE Romania'         ject Prinable String with 14 bytes length first above SEQUENCE and the SEQUENCE and the SEQUENCE and the first above SEQUENCE - the one with 38 bytes length (0x26) for subject           30         81 9F         SEQUENCE {         Another sequence within first sequence already opened and has 159 bytes length (0x9F)           30         0D         SEQUENCE {         Another sequence within first sequence already opened and has 159 bytes length           06         09         2a 86 48 86 F7 0D 01 01 01         OBJECT IDENTIFIER reaEncryption (1 2 840 113549 11 1)         OBJECT IDENTIFIER reaEncryption (1 2 840 113549 11 1)           05         00         NULL 1         Element NULL in ASIN. J BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.           03         81 8D         00 30 81 89 02 81 81 00 b6 31 f8 3c cf 87 50 8b 30 48 95 65 81 20 64 03 21 c4 43 44 4 a7 69 eb 46 7f fe7 ec 65 76 4b da c4 43 5 da 69 22 16 82 44 4 a7 69 eb 46 7f fe7 ec 65 76 4b da c4 43 4 a7 69 eb 46 7f fe7 ec 65 76 4b da c4 43 4 da 76 80 ec e5 5 sk 10 0 2 eq 20 52 4 7d 4 da f 81 a 0 6b dh fe 4 52 b da 79 21 ee 9d 4 29 ef 5 fc 2 2 ac 45 1 3 6 7e 00 59 eb 40 4 4 27 ef 5 fc 2 2 ac 45 1 5 6 76 2 b da 52 0 4d 2 27 ef 2 61 3 95 5 8 12 0 19 a 67 33 3e 4 d1 c5 af 68 96 ec e5 5 sk 4b 2c 29 95 24 7d 4 da f 1 a la 6b db fe 4 42 9 ff 6 2 2 ac 45 1 4 29 ef 5 fc 2 2 ac 45 1 4 29 ef 5 fc 2 2 ac 45 1 5 6 fc 2 ac 45 1 5 6 fc 2 ac 45 1 4 29 ef 5 fc 2 2 ac 45 1 5 6 fc 2 ac 5 2 ac 45 1 5 6 fc 2 ac 5 2 a				commonName (2 5 4 3)	
a       Image: Second Sec	13	0E	4F 52 41 4E 47 45 20	PrintableString	
30       81 9F			52 6f 6d 61 6e 69 61	'ORANGE Romania'	
30         81 9F         SEQUENCE { SEQUENCE { 0 penet and has 159 bytes length (0x9F) 0 penet and has 159 bytes length (0x9F)           30         0D         SEQUENCE { 0 penet and has 159 bytes length (0x9F)           30         0D         SEQUENCE { 0 penet and has 159 bytes length (0x9F)           06         09         2a 86 48 86 F7 0D 01 01 01         OBECT IDENTIFIER realizoption (1 2 840 113549 1 11)         OBECT IDENTIFIER realizoption (1 2 840 113549 1 11)           05         00         NULL 1 bytes length         Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.           03         81 8D         00 30 81 89 02 81 81 00 b6 31 f8 3c cf 87 50 8b 30 48 95 65 81         BIT STRING 0 unused 00 b6 31 68 3c cf 87 bits, encapsulates { 00 b6 31 68 3c cf 87 bits, encapsulates { 00 b6 31 68 3c cf 87 bits, encapsulates { 00 b6 32 cf 2 7 c2 56 76 4b bd ca d4 43 56 76 4b bd ca d4 43 57 8 10 90 9C 24 72 62 f6 13 58 R (2) 19 46 73 32 ed 41 c5 af 68 89 66 70 d0 10 5         Ead BIT STRING           30         0D         SEQUENCE { 40 20 30 10 00 1         Ead BIT ST				}	
30       81 9F       sEQUENCE {       Another sequence within first sequence already opened and has 159 bytes length (0x9F)         30       0D       SEQUENCE {       Another sequence within first two sequences already opened and has 13 bytes length         06       09       2a 86 48 86 F7 0D 01       OBJECT IDENTIFIER rataEncryption ::= rady opened and has 13 bytes length       (12 840 113549 11 1)         05       00       00 00 30 81 89 02 81 81       BIT STRING 0 unused       BIT STRING vith 141 bytes length for RSA         03       81 8D       00 05 31 18 3c cf 87 50 83 3d 48 95 65 81 2b 64 03 21 c4 43 4d 40 40 54 55 08 3d 44 95 65 81 2b 64 03 21 c4 43 4d 40 40 54 56 08 3d 44 95 65 81 2b 64 03 21 c4 43 4d 40 40 54 56 08 3d 44 95 65 81 4d 4d 55 6d 66 7f ff c7 ce 65 76 4b bb ca d4 43 5d 64 92 2f 7a 26 13 95 58 12 01 9a cf 33 3e d1 c5 af 6f 80 96 e5 54 0b 2c ff 9 52 24 7d 4f 1c 31 ac b6 bd fe 44 52 bd 64 12 c5 da 92 1c c9 d 42 9f 5f 62 2c c4 51 36 76 00 2f 7a 26 13 95 58 12 01 9a cf 33 3e d1 c5 af 6f 80 96 e5 54 0b 2c ef 9 55 24 7d 4f 1c 31 ac b6 bd fe 45 2b da 92 1c c9 d 42 9f 5f 62 2c c4 51 36 76 00 2f 7a 26 13 95 58 12 01 9a cf 33 2e d1 c5 af 6f 80 96 e5 54 0b 2c ef 9 55 24 7d 4f 1c 31 ac b6 bd fe 45 2b c4 17 2b (5a 86 9f 42 ef 9f 52 2c c4 51 36 76 00 01 2f 7a 26 13 2b (5a 7f 7a 2f 7a				}	
30         0D         SEQUENCE {         opened and has 13 bytes length (0x9F)           06         09         2a 86 48 86 F7 0D 01         00BJECT IDENTIFIER         ready opened and has 13 bytes length           06         09         2a 86 48 86 F7 0D 01         00BJECT IDENTIFIER         ready opened and has 13 bytes length           06         09         2a 86 48 86 F7 0D 01         00BJECT IDENTIFIER         ready opened and has 13 bytes length           05         00         NULL         840 113549 11 1)         BIT STRING view opened and has 13 bytes length for RSA           03         81 8D         00 30 81 89 02 81 81         BIT STRING 0 unused         BIT STRING with 141 bytes length for RSA           04         04 05 31 f8 3c cf 87         50 85 03 48 95 65 81         BIT STRING with 141 bytes length for RSA           05         00 b6 31 f8 3c cf 87         50 b6 30 48 95 65 81         BIT STRING with 141 bytes length for RSA           12 b6 40 32 1c 4 43 4d         a 71 9c 4b 67 ff 27 cc         bits, encapsulates {         Witk, encapsulates {           05         58 12 01 9a 67 33 3e         bits, encapsulates {         Witk and the tas bytes).           05 4 05 2 ff 22 ca 45 13 35         58 12 01 9a 67 33 3e         bits encapsulates {         Witk and the tas 3 bytes).           30         0D         20 20 20 cd c5 ef         20	20	81 OE		} SEQUENCE (	
30       0D       SEQUENCE {       Another sequence within first two sequences al-ready opened and has 13 bytes length         06       09       2a 86 48 86 F7 0D 01       OBJECT IDENTIFIER ready opened and has 13 bytes length       OBJECT IDENTIFIER ready opened and has 13 bytes length         07       01 01       (1 2 840 113549 11 1)       OBJECT IDENTIFIER ready opened and has 13 bytes length       OBJECT IDENTIFIER ready opened and has 13 bytes length         08       00       00       30 81 89 02 81 81       BIT STRING 0 unused       Element NULL in ASN.1 BER with 0 bytes length for RSA public Key. The second byte is the sequence (0x30), the third and fourth byte are the length of RSA public Key. The second byte is the sequence (0x30), the third and fourth bytes length. This is the length of the SA and dulus. After RSA modulus. After RSA modulus. Score the RSA public exponent. In this case, the RSA nublic exponent. In this case, the RSA public exponent.	50	01 91		SEQUENCE (	
06092a 86 48 86 F7 0D 01 01 01OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)ready opened and has 13 bytes length 0BJECT IDENTIFIER rsaEncryption ::= (iso(1) member-body(2) us(840) rsads(1(13549) pkcs(1) pkcs-1(1) rsaEncryption(1))0500NULL )Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.0381 8D00 30 81 89 02 81 81 00 b6 31 f8 3c cf 87 50 8b 30 48 95 65 81 2b 64 03 21 c4 43 4d a 76 9e 4b 67 ff cf cc 65 76 4b bd ca d4 43 56 da 69 2d 58 694 4e 0 66 42 2f 7e 2c 13 95 58 f2 01 9a 67 33 ac d1 c5 af 6f 89 6e c5 54 0b 2c 2f 7e 2d 13 95 58 f2 01 9a 67 33 ac d1 c5 af 6f 89 6e c5 54 0b 2c 2f 9 52 24 73 4f It c31 ac b6 bd fe 45 2b da 19 21 cc 9d 42 9f 5f 62 2e c4 51 36 6f 20 9d a52 04 d 2f c0 2e a2 02 ed c5 ef 02 03 01 00 01End BIT STRING another sequence within first two sequences al- ready opened and has 13 bytes length06092a 86 48 86 f7 0d 01 01 05OBJECT IDENTIFIER rsaWithShal (1 2 840 113549 11 5) (1 2 840 113549 11 5)End BIT STRING another sequence within first two sequences al- ready opened and has 13 bytes length05002a 86 48 86 f7 0d 01 01 05OBJECT IDENTIFIER rsaWithShal (1 2 840 113549 11 5) (1 2 840 113549 11 5)05002a 86 48 86 f7 0d 01 01 05OBJECT IDENTIFIER rsaWithShal (1 2 840 113549 11 5) (1 2 840 113549 11 5)0500NULL 2 8181OBJECT IDENTIFIER rsaWithShal (1 2 840 113549 11 5)0500NULL 2 8181Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.	30	0D		SEQUENCE {	
06       09       2a 86 48 86 F7 0D 01 01 01       OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)       OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)       OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)         05       00       NULL       Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.         03       81 8D       00 30 81 89 02 81 81 00 b6 31 f8 3c cf 87 50 8b 30 48 95 65 81 2b 64 03 21 c4 43 4d a 7 9e b4 b6 7f fe 7 ec 65 76 4b b6 ca 44 33 5d a6 92 d5 86 9d 4e 00 b6 2 d5 86 9d 4e 0 06 c c5 76 4b b6 ca 44 33 5d a6 92 d5 86 9d 4e 0 68 C response the RSA public key, 000189. The fith byte speci- fies an INTEGER of 0x181 bytes length. This is the length of the RSA modulus, After RSA modulus, comes the RSA public exponent is 0x01 0x01 0x01 (the last 3 bytes).         04       c5 af 6f 89 6e e5 54 0b 2e 79 52 z4 7d 4f lc 31 ac b6 bd fe 42 59 ff 62 2e c4 51 36 7e 00 59 eb b4 04 91 7f c4 f5 4c 2f bf 6a 8f 94 62 9d a5 20 4d 2f c0 2e a2 02 ed c5 ef 02 03 01 00 01       End BIT STRING         30       0D       SEQUENCE { Case 440 135549 1 15 01 05       SEQUENCE { Case 440 135549 1 15 01 000 08 45 29 49 7 c0				~- (	
05       00       NULL       Pkcs(1) pkcs-1(1) rsaEncryption(1)}         05       00       NULL       Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.         03       81 8D       00 30 81 89 02 81 81 00 63 11 83 cc f8 75 50 85 30 48 95 65 81 20 64 03 21 c4 43 4d a7 f9 eb 4b 67 ff e7 ec 65 75 4b bd ca d4 43 5d a6 92 d5 86 9d 4e 0b 4c eb 5b 08 33 41 96 eb 4c 65 5b 08 33 41 96 eb 4c 21 7e 2c 13 95 58 f2 01 9a 67 33 3e d1 c5 af 67 86 f6 80 6e 25 54 d0 b2 ef 9 95 24 7d 4f 1 c 31 ac b6 bd fe 45 2b da 92 1c e9 9d 42 9f 5f 62 2e c4 51 36 7e 00 59 eb b4 04 91 7f c4 f5 4c 2D fr 6a 8f 94 6e 20 4c 21 for 2c 10 fe a 8f 94 62 9d as 20 4d 2f c2 0 2e d5 ef 02 03 01 00 01       End BIT STRING         30       0D       SEQUENCE {       Another sequence within first two sequences already opened and has 13 bytes length. 10 20 10 50 fc 12 20 20 2d c5 ef 02 03 01 00 01         30       0D       2a 86 48 86 f7 0d 01 01 05 fc 12 EV IDENTIFIER rady opened and has 13 bytes length. 12 840 113549 1 1 5)       Signature algorithm identifier - sha1RSA = sha-11WithSha1 (1 2 840 113549 1 1 5)         05       00       81 81       00 08 d5 29 b4 97 e0       BIT STRING 0 unused BIT STRING with 129 bytes length - 1024 bits	06	09	2a 86 48 86 F7 0D 01	OBJECT IDENTIFIER	
0500NULL $\}$ Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.0381 8D00 30 81 89 02 81 81 00 b6 31 18 3c cf 87 50 8b 30 48 95 65 81 2b 64 03 21 c4 43 4d a 7f 9c b4 b6 7f fc 7c c 65 76 4b bd ca d4 43 5d a6 92 d5 86 9d 4e 0 b4 c c5 5b 08 33 41 9a 62 2f 7c 26 13 95 55 8f 20 19 a6 73 32 cd d4 fc 53 af 6f 89 6e e5 54 0b 2c ef 99 52 24 7d 4f fc 31 ac b6 bd fc 42 29 fs fc 22 ce 2 cd 51 36 7c 00 59 eb 40 4 91 7f c4 f5 4c 21 bf 6a 8f 94 c2 9d a5 20 4d 2f c0 2 a2 02 cd c5 efBIT STRING on SEQUENCE { end 42 29 fs fc 22 c2 cd 51 36 7c 00 59 eb 40 4 91 7f c4 f5 4c 21 bf 6a 8f 94 c2 9d a5 20 4d 2f c0 2 a2 02 cd c5 efEnd BIT STRING another sequence within first two sequences al- ready opened and has 13 bytes length of 02 03 01 00 01300DSEQUENCE { rawithSha1 (1 2 840 113549 11 5)End BIT STRING signature algorithm identifier - sha1RSA = sha- 1WithRSAEncryption OBJECT IDENTIFIER rawithSha1 (1 2 840 113549 11 5)0500NULL 3 81 8100 08 d5 29 b4 97 e0BIT STRING 0 unused0381 8100 08 d5 29 b4 97 e0BIT STRING 0 unusedBIT STRING with 129 bytes length - 1024 bits-			01 01	rsaEncryption	
03         81 8D         00 30 81 89 02 81 81 00 b6 31 f8 3c cf 87 50 80 30 48 95 65 81 2b 64 03 21 c4 43 4d a7 f9 eb 4b 67 ff e7 ec 65 76 4b bd ca d4 43 5d a6 92 d5 86 9d 4e 0 b6 c c5 50 83 34 1 9a 62 2f 7e 26 13 95 58 f2 01 9a 67 33 3c d1 c5 af 6f 89 6e e5 54 0b 2c a4 92 de c5 ef 153 61 7c 00 59 eb 4b 04 4f 1c 51 ac b6 b6 de 45 2b da f9 21 cc 9d 42 9f 5f 62 2c c4 51 36 7e 00 59 eb 4b 04 91 7f c4 f5 4c 2f bf 6a 8f 94 62 0d a 20 de c5 ef 102 03 01 00 01         End BIT STRING           30         0D         5EQUENCE { 102 30 100 01         End BIT STRING           30         0D         SEQUENCE { 102 30 10 00 11         SEQUENCE { 12 840 113549 11 5)           30         0D         2a 86 48 86 f7 0d 01 01 05         OBJECT IDENTIFIER raWithShal (1 2 840 113549 11 5)         Signature algorithm identifier - shal RSA = sha- ready opened and has 13 bytes length.           06         09         2a 86 48 86 f7 0d 01 01 05         OBJECT IDENTIFIER raWithShal (1 2 840 113549 1 1 5)         Signature algorithm identifier - shal RSA = sha- ready opened and has 13 bytes length.           06         09         2a 86 48 86 f7 0d 01 01 05         OBJECT IDENTIFIER raWithShal (1 2 840 113549 1 1 5)         Signature algorithm identifier - shal RSA = sha- ready opened and has 13 bytes length.           05         00         NULL         Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.           03         81 81         00 08 d5 29 b4 97 e0         BIT STRING 0 unused         BIT STRING with 129 bytes length - 1024 bits - <td></td> <td></td> <td></td> <td>(1 2 840 113549 1 1 1)</td> <td><pre>pkcs(1) pkcs-1(1) rsaEncryption(1)}</pre></td>				(1 2 840 113549 1 1 1)	<pre>pkcs(1) pkcs-1(1) rsaEncryption(1)}</pre>
03         81 8D         00 30 81 89 02 81 81 00 b6 31 f8 3c cf 87 50 80 30 48 95 65 81 2b 64 03 21 c4 43 4d a7 f9 eb 4b 67 ff e7 ec 65 76 4b bd ca d4 43 5d a6 92 d5 86 9d 4e 0 b6 c c5 50 83 34 1 9a 62 2f 7e 26 13 95 58 f2 01 9a 67 33 3c d1 c5 af 6f 89 6e e5 54 0b 2c a4 92 de c5 ef 153 61 7c 00 59 eb 4b 04 4f 1c 51 ac b6 b6 de 45 2b da f9 21 cc 9d 42 9f 5f 62 2c c4 51 36 7e 00 59 eb 4b 04 91 7f c4 f5 4c 2f bf 6a 8f 94 62 0d a 20 de c5 ef 102 03 01 00 01         End BIT STRING           30         0D         5EQUENCE { 102 30 100 01         End BIT STRING           30         0D         SEQUENCE { 102 30 10 00 11         SEQUENCE { 12 840 113549 11 5)           30         0D         2a 86 48 86 f7 0d 01 01 05         OBJECT IDENTIFIER raWithShal (1 2 840 113549 11 5)         Signature algorithm identifier - shal RSA = sha- ready opened and has 13 bytes length.           06         09         2a 86 48 86 f7 0d 01 01 05         OBJECT IDENTIFIER raWithShal (1 2 840 113549 1 1 5)         Signature algorithm identifier - shal RSA = sha- ready opened and has 13 bytes length.           06         09         2a 86 48 86 f7 0d 01 01 05         OBJECT IDENTIFIER raWithShal (1 2 840 113549 1 1 5)         Signature algorithm identifier - shal RSA = sha- ready opened and has 13 bytes length.           05         00         NULL         Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.           03         81 81         00 08 d5 29 b4 97 e0         BIT STRING 0 unused         BIT STRING with 129 bytes length - 1024 bits - <td>05</td> <td></td> <td></td> <td></td> <td></td>	05				
03         81 8D         00 30 81 89 02 81 81 00 b6 31 f8 3 c f 87 50 8b 30 48 95 65 81 2b 64 03 21 c4 43 4d a 7 f9 eb 4b 67 ff e7 ec 65 7 64 bb dc ad 44 3 56 57 64 bb dc ad 44 3 56 af 69 2 de 58 69 d4 e 0 bb 4c cb 5b 08 33 41 9a 62 2f 7e 26 13 95 58 f2 01 9a 67 33 3e d1 c5 af 68 96 6e 5 54 0b 2e f9 95 24 7d 4f 1c 31 ac b6 bd fe 45 2b da f9 21 ce 9d 42 9f 5f 62 2e c4 51 36 7e 00 59 eb b4 04 91 7f c4 f5 4c 2f 16 fa 8 8f 94 62 9d a5 20 4d 2f c0 2e a2 02 ed c5 ef 02 03 01 00 01         End BIT STRING           30         0D         2         86 48 86 f7 0d 01 05         0BJECT IDENTIFIER rasWithSha1 (1 2 840 113549 11 5)         End BIT STRING           30         0D         2         86 48 86 f7 0d 01 05         0BJECT IDENTIFIER rasWithSha1 (1 2 840 113549 11 5)         End BIT STRING           30         00         2         86 48 86 f7 0d 01 05         0BJECT IDENTIFIER rasWithSha1 (1 2 840 113549 11 5)         Signature algorithm identifier - sha1RSA = sha- ready opened and has 13 bytes length           30         00         2         86 48 86 f7 0d 01 0 5         OBJECT IDENTIFIER rasWithSha1 (1 2 840 113549 11 5)         Signature algorithm identifier - sha1RSA = sha- ready opened and has 13 bytes length           30         00         2         86 48 86 f7 0 d01 0 5         DBJECT IDENTIFIER rasWithSha1 (1 2 840 113549 1 1 5)         Signature(5) rasWithSha1 (1 2 840 113549 1 1 5)           30         00         3         81 81         00 08 d5 29 b4 97 e0 </td <td>05</td> <td>00</td> <td></td> <td>NULL</td> <td></td>	05	00		NULL	
03         81 8D         00 30 81 89 02 81 81 00 b6 31 18 3c cf 87 50 8b 30 48 95 65 81 2b 64 03 21 c4 43 4d a7 f9 eb 4b 67 ff e7 ec 65 76 4b bd ca d4 43 5d a6 92 d5 86 9d 4e 10 b4 cc b5 b0 83 34 1 9a 62 2f 7e 2c 13 95 58 f2 01 9a 67 33 3c d1 c5 af 6f 89 6e e5 54 0b 2c a 42 7b 6a d1 c5 af 6f 89 6e e5 54 0b 2c 19 95 24 7d 4f 1c 31 ac b6 bd fe 45 2b da f9 21 ec 9d 42 9f 5f 62 2e c4 51 36 7e 00 59 eb 40 40 91 7f c4 f5 4c 2f bf 6a 8f 94 62 9d a5 20 4d 2f c0 2e a2 02 ed c5 ef 02 03 01 00 01         End BIT STRING         BIT STRING           30         0D         38 64 88 6f 70 d0 11 01 05         OBJECT IDENTIFIER raWithShal (1 2 840 113549 11 5)         End BIT STRING           30         0D         2a 86 48 86 f7 0d 01 01 05         OBJECT IDENTIFIER raWithShal (1 2 840 113549 11 5)         End BIT STRING           30         00         2a 86 48 86 f7 0d 01 01 05         OBJECT IDENTIFIER raWithShal (1 2 840 113549 11 5)         Signature algorithm identifier - sha1RSA = sha- isignature(5)           00         81 81         00 08 d5 29 b4 97 c0         BIT STRING 0 unused BIT STRING with 129 bytes length - 1024 bits -				}	
00         b6         31         f8         3c         cf87           50         8b         30         48         95         68         11           2b         64         03         2t         44         3d         of 85         public Key. The second byte is the sequence           07         97         9e         b4         67         ff 7e         cc         field         of 85         public Key. 0x181         bytes are the length         of 85         public key. 0x181         bytes are the length         of 85         public key. 0x181         bytes are the length         of 85         public key. 0x181         bytes are the length         of 85         public key. 0x181         bytes are the length         of 85         public key. 0x181         bytes are the length         of 85         public key. 0x181         bytes protecties an INTEGER         fies an INTEGER	03	81 8D	00 30 81 89 02 81 81	BIT STRING 0 unused	
30       0D       25 64 03 21 c4 43 4d a       (0x30), the third and fourth byte are the length of RSA public key, 0x0189. The fith byte specifies an INTEGER of 0x181 bytes length. This is the length of the RSA modulus. After RSA modulus, comes the RSA public exponent. In this case, the RSA public exponent is 0x01 0x01 0x01 (the last 3 bytes).         9       62 27 7 26 13 95       0x01 (the last 3 bytes).         58 f2 01 9a 67 33 3e       0x01 (the last 3 bytes).         10 5 4c cb 5b 08 33 41       0x01 (the last 3 bytes).         9       62 27 7 26 13 95         58 f2 01 9a 67 33 3e       0x01 (the last 3 bytes).         4f ic 31 ac b6 bd fe       45 2b da f9 21 ec 9d         42 9f 5f 62 2e c4 51       56 20 9b b4 04         91 7f c4 f5 4c 2f bf 6a       8f 94 62 9d as 20 4d         91 7f c4 f5 4c 2f bf 6a       8f 94 62 9d as 20 4d         92 7c 02 ea 20 2e dc 5e f       02 30 10 001         91       30       0D         92       2a 86 48 86 f7 0d 01       OBJECT IDENTIFIER rsaWithSha1         (1 2 840 113549 11 5)       Signature algorithm identifier - sha1RSA = sha-1WithRSAEncryption OBJECT IDENTIFIER rsaWithSha1         (1 2 840 113549 11 5)       Signature algorithm identifier - sha1RSA = sha-1WithRSAEncryption OBJECT IDENTIFIER rsaWithSha1         (1 2 840 113549 11 5)       Signature algorithm identifier - sha1RSA = sha-1WithRSAEncryption OBJECT IDENTIFIER rsignature(5) <td< td=""><td></td><td></td><td></td><td></td><td></td></td<>					
a7 f9 eb 4b 67 ff e7 ec       fes an INTEGER of 0x181 bytes length. This is         65 76 4b bd ca d4 43       fes an INTEGER of 0x181 bytes length. This is         bd a6 92 d5 86 9d 4e       modulus, comes the RSA public exponent. In         0b 4c cb 5b 08 33 41       modulus, comes the RSA public exponent. In         9a 62 2f 7e 26 13 95       ss         58 f2 01 9a 67 33 3e       01 c5 af 6f 89 6e e5         d1 c5 af 6f 89 6e e5       st         54 0b 2e f9 95 24 7d       st         4f 1c 31 ac b6 bd fe       st         42 9f 5f 62 ce 45 1       st         36 7e 00 59 eb b4 04       91 7f c4 f5 4c 2f bf 6a         8f 94 62 9d a5 20 4d       2f c0 2e ac 20 2ed         90 0D       SEQUENCE {         30       0D       SEQUENCE {         01 05       SEQUENCE {         12 840 113549 11 5)       Signature algorithm identifier - sha1RSA = sha-1WithRSAEncryption 0BJECT IDENTIFIER         ready opened and has 13 bytes length       signature(5)         05       00       NULL       Element NULL in ASN.1 BER with 0 bytes length.         112 840 113549 11 5)       signature(5)       signature(5)         05       00       NULL       Element NULL in ASN.1 BER with 0 bytes length.         05       00       NULL       Element NU					
65 76 4b bd ca d4 43       the length of the RSA modulus. After RSA modulus. comes the RSA public exponent. In this case, the RSA public exponent. In this case, the RSA public exponent is 0x01 0x01         9a 62 27 7e 26 13 95       58 f2 01 9a 67 33 3e       0x01 (the last 3 bytes).         65 76 4b bd ca d4 43       58 f2 01 9a 67 33 2e       0x01 (the last 3 bytes).         61 c5 af 6f 89 6e e5       54 0b 2e f9 95 24 7d       44 f1 c 31 ac b6 bd fe         41 c 31 ac b6 bd fe       45 2b da f9 21 ec 9d       44 f1 c 31 ac b6 bd fe         42 9f 5f 62 2e c4 51       36 7e 00 59 eb b4 04       91 7f c4 f5 4c 2f bf 6a         8f 94 62 9d a5 20 4d       2f c0 2e a 20 2e dc 5e f0 20 30 10 00 11       End BIT STRING         30       0D       SEQUENCE {       Another sequence within first two sequences already opened and has 13 bytes length         06       09       2a 86 48 86 f7 0d 01       OBJECT IDENTIFIER rsaWithSha1 (1 2 840 113549 1 1 5)       Signature algorithm identifier - sha1RSA = sha-11WithRSAEncryption OBJECT IDENTIFIER rsaWithSha1 (1 2 840 113549 1 1 5)         07       00       NULL       Element NULL in ASN.1 BER with 0 bytes length.         08       81 81       00 08 d5 29 b4 97 e0       BIT STRING 0 unused       BIT STRING with 129 bytes length - 1024 bits -					
5d a6 92 d5 86 9d 4e       modulus, comes the RSA public exponent. In this case, the RSA public exponent is 0x01 0x01         9a 62 2f 7e 26 13 95       58 f2 01 9a 67 33 3e       modulus, comes the RSA public exponent is 0x01 0x01         9a 62 2f 7e 26 f3 95       58 f2 01 9a 67 33 3e       nthis case, the RSA public exponent is 0x01 0x01         9a 62 2f 7e 26 f3 95       58 f2 01 9a 67 33 3e       nthis case, the RSA public exponent is 0x01 0x01         9a 62 2f 7e 26 f3 95       58 f2 01 9a 67 33 3e       nthis case, the RSA public exponent is 0x01 0x01         9a 61 c5 af 6f 89 6e e5       54 0b 2e f9 95 24 7d       nthis case, the RSA public exponent is 0x01 0x01         4f 1c 31 ac b6 bd fe       45 2b da f9 21 ec 9d       nthis case, the RSA public exponent is 0x01 0x01         42 9f 5f 62 2e c4 51       36 7e 00 59 eb b4 04       91 7f c4 f5 4c 2f bf 6a         8f 94 62 9d a5 20 4d       2f c0 2e a2 02 ed c5 ef       nthis requence within first two sequences already opened and has 13 bytes length         30       0D       Signature algorithm identifier - sha1RSA = sha-11WithRSAEncryption OBJECT IDENTIFIER         rsaWithSha1       (1 2 840 113549 1 1 5)       Signature algorithm identifier - sha1RSA = sha-11WithRSAEncryption OBJECT IDENTIFIER         rsignature(5)       00       NULL       Element NULL in ASN.1 BER with 0 bytes length.         05       00       NULL       Element NULL in ASN.1 BER with 0 bytes length.					
0b       4c cb 5b 08 33 41       9a 62 2f 7e 26 13 95       58 f2 01 9a 67 33 3e       this case, the RSA public exponent is 0x01 0x01         0x01 (the last 3 bytes).       58 f2 01 9a 67 33 3e       0x01 (the last 3 bytes).       0x01 (the last 3 bytes).         54 0b 2e f9 95 24 7d       4f 1c 31 ac b6 bd fe       45 2b da f9 21 ec 9d       42 9f 5f 62 2e c4 51       56 76 00 59 eb 404         91 7f c4 f5 4c 2f bf 6a       8f 94 62 9d a5 20 4d       2f c0 2e a2 02 ed c5 ef       02 ed c5 ef       02 ed c5 ef         30       0D       30 0D       30 0D       SEQUENCE {       Another sequence within first two sequences already opened and has 13 bytes length         06       09       2a 86 48 86 f7 0d 01       0BJECT IDENTIFIER rawithSha1       Signature algorithm identifier - sha1RSA = sha-11WithRSAEncryption OBJECT IDENTIFIER rawithSha1       Signature algorithm identifier - sha1RSA = sha-11WithRSAEncryption OBJECT IDENTIFIER rawithSha1         06       09       2a 86 48 86 f7 0d 01       0BJECT IDENTIFIER rawithSha1       Signature algorithm identifier - sha1RSA = sha-11WithRSAEncryption OBJECT IDENTIFIER rawithSha1         01 05       12 840 113549 11 5)       ::= (iso(1) member-body(2) us(840) rsad-si(113549) pkcs(1) pkcs-1(1) sha1-with-rsa-signature(5)}         05       00       NULL       Element NULL in ASN.1 BER with 0 bytes length.         03       81 81       00 08 d5 29 b4 97 e0       BIT STRING 0 unused					
9a 62 2f 7e 26 13 95 58 f2 01 9a 67 33 3e d1 c5 af 6f 89 6e e5 54 0b 2e f9 95 24 7d 4f 1c 31 ac b6 bd fe 45 2b da f9 21 ec 9d 42 9f 5f 62 2e c4 51 36 7e 00 59 eb b4 04 91 7f c4 f5 4c 2f bf 6a 8f 94 62 9d a5 20 4d 2f c0 2e a2 02 ed c5 ef 02 03 01 00 01       SEQUENCE {       End BIT STRING         30       0D       2a 86 48 86 f7 0d 01 01 05       OBJECT IDENTIFIER rsaWithSha1 (1 2 840 113549 11 5)       Signature algorithm identifier - sha1RSA = sha- isignature(5)}         05       00       NULL       2k0 113549 11 5)       Element NULL in ASN.1 BER with 0 bytes is ending.         03       81 81       00 08 d5 29 b4 97 e0       BIT STRING 0 unused       BIT STRING with 129 bytes length - 1024 bits -					· · ·
58 f2 01 9a 67 33 3e d1 c5 af 6f 89 6e e5 54 0b 2e f9 95 24 7d 4f 1c 31 ac b6 bd fe 45 2b da f9 21 ec 9d 42 9f 5f 62 2e c4 51 36 7e 00 59 eb b4 04 91 7f c4 f5 4c 2l bf 6a 8f 94 62 9d a5 20 4d 2f c0 2e a2 02 ed c5 ef 02 03 01 00 01         End BIT STRING           30         0D         }         SEQUENCE {         Another sequence within first two sequences al- ready opened and has 13 bytes length           06         09         2a 86 48 86 f7 0d 01 01 05         OBJECT IDENTIFIER rsaWithSha1 (1 2 840 113549 11 5)         Signature algorithm identifier - sha1RSA = sha- 1WithRSAEncryption OBJECT IDENTIFIER signature(5)}           05         00         NULL         Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.           03         81 81         00 08 d5 29 b4 97 e0         BIT STRING 0 unused         BIT STRING with 129 bytes length - 1024 bits -					
d1 c5 af 6f 89 6e e5       54 0b 2e f9 95 24 7d       4f 1c 31 ac b6 bd fe         4f 1c 31 ac b6 bd fe       45 2b da f9 21 ec 9d       42 9f 5f 62 2e c4 51         36 7e 00 59 eb b4 04       91 7f c4 f5 4c 2f bf 6a       8f 94 62 9d a5 20 4d         2f c0 2e a2 02 ed c5 ef       02 03 01 00 01       8EQUENCE {         30       0D       SEQUENCE {       Another sequence within first two sequences already opened and has 13 bytes length         06       09       2a 86 48 86 f7 0d 01       OBJECT IDENTIFIER rsaWithSha1       Signature algorithm identifier - sha1RSA = sha-11WithRSAEncryption OBJECT IDENTIFIER rsaWithSha1         01 05       12 840 113549 1 1 5)       ::= {iso(1) member-body(2) us(840) rsad-si(113549) pkcs(1) pkcs-1(1) sha1-with-rsa-signature(5)}         05       00       NULL       Element NULL in ASN.1 BER with 0 bytes length.         03       81 81       00 08 d5 29 b4 97 e0       BIT STRING 0 unused       BIT STRING with 129 bytes length - 1024 bits -					onor (the last o by es).
4f 1c 31 ac b6 bd fe 45 2b da f9 21 ec 9d 42 9f 5f 62 2e c4 51 36 7e 00 59 eb b4 04 91 7f c4 f5 4c 2f bf 6a 8f 94 62 9d a5 20 4d 2f c0 2e a2 02 ed c5 ef 02 03 01 00 01End BIT STRING300D}End BIT STRING300DSEQUENCE { 01 05Another sequence within first two sequences al- ready opened and has 13 bytes length06092a 86 48 86 f7 0d 01 01 05OBJECT IDENTIFIER rsaWithSha1 (1 2 840 113549 11 5)Signature algorithm identifier - sha1RSA = sha- 1WithRSAEncryption OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840) rsad- signature(5)}0500NULL PNULL PElement NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.0381 8100 08 d5 29 b4 97 e0BIT STRING 0 unusedBIT STRING with 129 bytes length - 1024 bits -					
45 2b da f9 21 ec 9d 42 9f 5f 62 2e c4 51 36 7e 00 59 eb b4 04 91 7f c4 f5 4c 2f bf 6a 8f 94 62 9d a5 20 4d 2f c0 2e a2 02 ed c5 ef 02 03 01 00 01End BIT STRING300D}End BIT STRING300D2a 86 48 86 f7 0d 01 01 05OBJECT IDENTIFIER rsaWithSha1 (1 2 840 113549 1 1 5)Signature algorithm identifier - sha1RSA = sha- 1WithRSAEncryption OBJECT IDENTIFIER signature(5)}0500NULL PElement NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.0381 8100 08 d5 29 b4 97 e0BIT STRING 0 unusedBIT STRING with 129 bytes length - 1024 bits -			54 0b 2e f9 95 24 7d		
42 9f 5f 62 2e c4 51 36 7e 00 59 eb b4 04 91 7f c4 f5 4c 2f bf 6a 8f 94 62 9d a5 20 4d 2f c0 2e a2 02 ed c5 ef 02 03 01 00 01End BIT STRING300D}End BIT STRING300DSEQUENCE { 10 05Another sequence within first two sequences al- ready opened and has 13 bytes length06092a 86 48 86 f7 0d 01 01 05OBJECT IDENTIFIER rsaWithSha1 (1 2 840 113549 11 5)Signature algorithm identifier - sha1RSA = sha- 1WithRSAEncryption OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840) rsad- signature(5)}0500NULL 2Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.0381 8100 08 d5 29 b4 97 e0BIT STRING 0 unusedBIT STRING with 129 bytes length - 1024 bits -					
36 7e 00 59 eb b4 04 91 7f c4 f5 4c 2f bf 6a 8f 94 62 9d a5 20 4d 2f c0 2e a2 02 ed c5 ef 02 03 01 00 01Image: Second Seco					
91 7f c4 f5 4c 2f bf 6a 8f 94 62 9d a5 20 4d 2f c0 2e a2 02 ed c5 ef 02 03 01 00 01End BIT STRING300D}End BIT STRING300DSEQUENCE { - ready opened and has 13 bytes length06092a 86 48 86 f7 0d 01 01 05OBJECT IDENTIFIER rsaWithSha1 (1 2 840 113549 1 1 5)Signature algorithm identifier - sha1RSA = sha- 1WithRSAEncryption OBJECT IDENTIFIER signature(5)}0500NULL - ready opened and has 13 bytes lengthElement NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.0381 8100 08 d5 29 b4 97 e0BIT STRING 0 unusedBIT STRING with 129 bytes length - 1024 bits -					
8f 94 62 9d a5 20 4d 2f c0 2e a2 02 ed c5 ef 02 03 01 00 01End BIT STRING300D}End BIT STRING300DSEQUENCE {Another sequence within first two sequences al- ready opened and has 13 bytes length06092a 86 48 86 f7 0d 01 01 05OBJECT IDENTIFIER rsaWithSha1 (1 2 840 113549 1 1 5)Signature algorithm identifier - sha1RSA = sha- 1WithRSAEncryption OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840) rsad- si(113549) pkcs(1) pkcs-1(1) sha1-with-rsa- signature(5)}0500NULL PElement NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.0381 8100 08 d5 29 b4 97 e0BIT STRING 0 unusedBIT STRING with 129 bytes length - 1024 bits -					
2f c0 2e a2 02 ed c5 ef 02 03 01 00 01End BIT STRING300D}End BIT STRING300DSEQUENCE {Another sequence within first two sequences al- ready opened and has 13 bytes length06092a 86 48 86 f7 0d 01 01 05OBJECT IDENTIFIER rsaWithSha1 (1 2 840 113549 1 1 5)Signature algorithm identifier - sha1RSA = sha- 1WithRSAEncryption OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840) rsad- si(113549) pkcs(1) pkcs-1(1) sha1-with-rsa- signature(5)}0500NULL PElement NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.0381 8100 08 d5 29 b4 97 e0BIT STRING 0 unusedBIT STRING with 129 bytes length - 1024 bits -					
02 03 01 00 01End BIT STRING300DSEQUENCE {Another sequence within first two sequences already opened and has 13 bytes length06092a 86 48 86 f7 0d 01 01 05OBJECT IDENTIFIER rsaWithSha1 (1 2 840 113549 1 1 5)Signature algorithm identifier - sha1RSA = sha- 1WithRSAEncryption OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840) rsad- si(113549) pkcs(1) pkcs-1(1) sha1-with-rsa- signature(5)}0500NULL PElement NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.0381 8100 08 d5 29 b4 97 e0BIT STRING 0 unusedBIT STRING with 129 bytes length - 1024 bits -					
30       0D       SEQUENCE {       Another sequence within first two sequences already opened and has 13 bytes length         06       09       2a 86 48 86 f7 0d 01 01 05       OBJECT IDENTIFIER rsaWithSha1       Signature algorithm identifier - sha1RSA = sha- 1WithRSAEncryption OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsad- si(113549) pkcs(1) pkcs-1(1) sha1-with-rsa- signature(5) }         05       00       NULL }       Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.         03       81 81       00 08 d5 29 b4 97 e0       BIT STRING 0 unused       BIT STRING with 129 bytes length - 1024 bits -					
06092a 86 48 86 f7 0d 01 01 05OBJECT IDENTIFIER rsaWithSha1 (1 2 840 113549 1 1 5)Signature algorithm identifier - sha1RSA = sha- 1WithRSAEncryption OBJECT IDENTIFIER isi(113549) pkcs(1) pkcs-1(1) sha1-with-rsa- signature(5)}0500NULL PElement NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.0381 8100 08 d5 29 b4 97 e0BIT STRING 0 unusedBIT STRING with 129 bytes length - 1024 bits -				}	
06092a 86 48 86 f7 0d 01 01 05OBJECT IDENTIFIER rsaWithSha1 (1 2 840 113549 1 1 5)Signature algorithm identifier - sha1RSA = sha- 1WithRSAEncryption OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840) rsad- signature(5)}0500NULL -Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.0381 8100 08 d5 29 b4 97 e0BIT STRING 0 unusedBIT STRING with 129 bytes length - 1024 bits -	30	0D		SEQUENCE {	
01 05       rsaWithSha1       1WithRSAEncryption OBJECT IDENTIFIER         (1 2 840 113549 1 1 5)       ::= {iso(1) member-body(2) us(840) rsad-si(113549) pkcs(1) pkcs-1(1) sha1-with-rsasignature(5)}         05       00       NULL       Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.         03       81 81       00 08 d5 29 b4 97 e0       BIT STRING 0 unused       BIT STRING with 129 bytes length - 1024 bits -	06	00			
05       00       NULL       Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.         03       81 81       00 08 d5 29 b4 97 e0       BIT STRING 0 unused       BIT STRING with 129 bytes length - 1024 bits -	06	09			
05       00       NULL       Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.         03       81 81       00 08 d5 29 b4 97 e0       BIT STRING 0 unused       BIT STRING with 129 bytes length - 1024 bits -			01 05		
05       00       NULL       Element NULL in ASN.1 BER with 0 bytes         05       00       Image: Second se				(1207011337113)	
05       00       NULL       Element NULL in ASN.1 BER with 0 bytes length. So, NULL 2 bytes and the SEQUENCE is ending.         03       81 81       00 08 d5 29 b4 97 e0       BIT STRING 0 unused       BIT STRING with 129 bytes length - 1024 bits -					
03     81 81     00 08 d5 29 b4 97 e0     BIT STRING 0 unused     BIT STRING with 129 bytes length - 1024 bits -	05	00		NULL	
03 81 81 00 08 d5 29 b4 97 e0 BIT STRING 0 unused BIT STRING with 129 bytes length - 1024 bits -				}	length. So, NULL 2 bytes and the SEQUENCE
CI OD IG DI IU UU CE D3   bits, encapsulates {   for digital signature of Certificates Authority.	03	81 81			
			cī 6b fd bi f0 00 ce b3	bits, encapsulates {	for digital signature of Certificates Authority.

	89 3c 7a 7c f2 95 23			This certificate is self signed and the signature is	
	1c a2 56 ca 09 04 b6			generated with itself private key. In this case all	
	74 d3 f5 84 15 ff cc 03			the previous fields from the certificate has been	
	1e 2a b4 69 ac 83 42			concatenated and signed with the private key	
	03 c7 b6 d1 a0 e1 51			and can be verified with itself public key.	
	8c eb a1 3f 32 53 89				
	ee 5b ae 72 6e a6 16				
	ac 88 77 38 23 cb bc				
	8e 15 7f 36 4f 8c 92				
	4d e9 53 96 0b 08 c9				
	cd 92 fd 46 fc 9e 05				
	11 bd ba cb 6f 30 53				
	e8 94 48 45 84 68 c8				
	ba c1 57 89 75 a1 32				
	f2 35 01 1b e7 17 87				
	9c b7 6d 8b eb af 7a				
	9a				
		}		End BIT STRING	
		}		End SEQUENCE with 0x81 0x9F length	
		}		End SEQUENCE with 82 0x01 0xC3 length	
So from the n	revious table it is o	byious the	OBI	ECT IDENTIFIER countryName (2.5.4.6)	
· •				tableString 'RO'	
following for the X509 v3 self signed certifi-			}		
cate:			}		
■ the serial m	umber is 11835482	260 (0x46)	SET {		
		200 (0740	SEQUENCE {		
0x8B 0x83 0x6	04 nex);		$ODIECT IDENTIFIED \qquad N = (2.5.4.2)$		

 the certificate is signed with RSA over the SHA-1 hash algorithm;

- the issuer's distinguished name is CN = ORANGE Romania; C = RO;
- the subject's distinguished name is CN = ORANGE Romania; C = RO;

• the issuer and the subject are the same, so the certificate is self signed;

• the certificate was issued on July 04, 2007 11:24:20 and will expire on October 02, 2007 11:24:20;

• the certificate contains a 1024 bit RSA public key;

• the certificate contains the signature of all the fields in last section

The obtained ASN.1 Notation for the certificate from the previous listing of the self signed certificate is [4]:

```
SEQUENCE {

SEQUENCE {

[0] {

INTEGER 2

}

INTEGER 1183548260

SEQUENCE {

OBJECT IDENTIFIER

rsaWithSha1(1.2.840.113549.1.1.5)

NULL

}

SEQUENCE {

SET {

SEQUENCE {
```

```
OBJECT IDENTIFIER commonName (2.5.4.3)
   PrintableString 'ORANGE Romania'
  }
 }
SEQUENCE {
 UTCTime '070704112420Z'
 UTCTime '071002112420Z'
SEQUENCE {
 SET {
   SEQUENCE {
      OBJECT IDENTIFIER countryName (2.5.4.6)
      PrintableString 'RO'
   }
  }
  SET {
   SEQUENCE {
      OBJECT IDENTIFIER commonName (2.5.4.3)
      PrintableString 'ORANGE Romania'
    }
  }
SEQUENCE {
 SEQUENCE {
  OBJECT IDENTIFIER rsaEncryption (1.2.840.11354
   9.1.1.1)
  NULL
 BIT STRING 0 unused bits, encapsulates { //public
   key}
 SEQUENCE {
  OBJECT IDENTIFIER rsaWithSha1 (1.2.840.113549
    .1.1.5)
  NULL
 BIT STRING 0 unused bits, encapsulates { //signat
    ure }
```

The systems for obtaining public keys based on certificates are simple and cheap to implement, due to an important characteristic of the digital certificates: the certificates may be distributed without requiring protection through the average security systems (certification, integrity and confidentiality). This because the public key should not be kept secret; thus, the digital certificate that includes it is not secret. There are no requirements for certification or integrity, because the certificate self-protects (the digital signature of AC in the certificate ensures its certification, as well as its integrity).

Consequently, the digital certificates may be distributed and moved by unsecured communication liaisons: by unsecured file servers, by systems of unsecured folders and/or communication protocols that do not endure the security.

#### 3. Protocols for Certificates Management

The interaction between the components of a public key infrastructure requires the existence of some protocols for the certificates management. The elements involved in the PKI management are:

• the subject of a certificate, that may be a person or an application and represents the final entity (EE – End Entity);

• Certification Authority – CA, that starts a digital certificate, coupling the identity of an user with his public key and certifies this association;

• AI (Registration Authority – RA) – certification of the persons, name deliverance key generation.

The protocols between the mentioned above are used for the following scopes:

• Establishing CA: when a new CA is established, there must be taken certain steps, such as generating the initial list of revoked certificates or the export of the CA's public key.

• Initializing the final entity: involves obtaining the public key of the root CA and the information request about the PKI management at the level of the final entity

• Certification:

➤ Initial registration/certification – when an end entity becomes known to a Certification Authority (CA). After this process CA generates one or more certificates for that end entity.

Establishing a pair of keys – it is necessary that each pair of keys to be regularly changed and a new certificate to be issued.

➢ Updating a certificate – when a certificate expires, it must be update.

Changing the CA's pair of keys.

Request for crossed certificates – when a certification authority certifies another certification authority.

Updating some crossed certificates

• Publishing of a certificate or list of revoked certificates: involves storing a certificate or a list of revoked certificates where everybody may have access (for instance such a protocol is LDAP).

• Restoration of a pair of keys: when an end entity loses its private key and wishes restoration, if previously RA or CA saved this key.

• Revoking a certificate: when an end entity wishes to revoke (cancel) a certificate, operation that involves a revocation demand and implicitly the update of the list of revoked certificates (CRL – Certificate Revocation Lists).

It is not necessary that these operations should be executed on-line, also existing offline methods for their fulfillment. It is impossible to find a Certification Authority which to issue certificates for all the owners of pairs of public/ private keys in the world, because it is not practical that all the users in the world trust a single organization or company in what concerns their secret communications. That is why is accepted the idea of the multiple Certification Authorities. Moreover, a single user can have certificates issued by different CAs, for different types of secure communications he wants to establish. As well, practically it is not possible to presume that a users of the public key infrastructure (PKI) already holds the public key of a certain certification authority, CA<sub>1</sub>, which issued a certificate for an entity with whom that user wants to secure communicate. Although, in order to obtain the public key of that  $CA_1$ , the user may use another certificate, i.e. a certificate issued for that  $CA_1$  by another certification authority,  $CA_2$ , whose public key is hold in a secure way by the user. So, the procedure is recursively applied and a user may obtain the public keys for a constantly larger number of Certification Authorities and, correspondingly, the public keys for a constantly larger number of other users. This leads to a general pattern, called certification chain of certification path, on which are based all the main present systems for public keys distribution.

The model of building the certification chain is reproduced in figure 8.

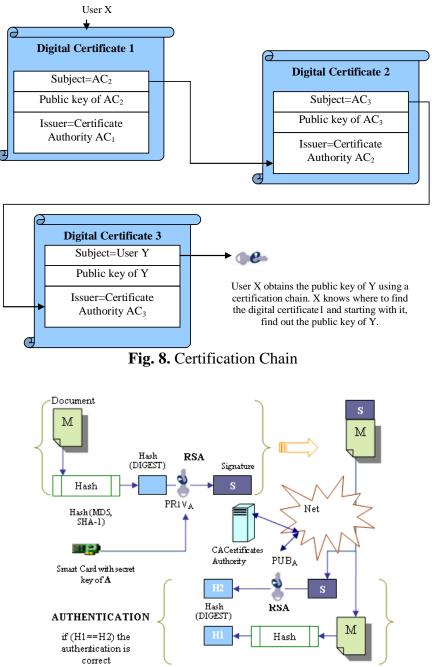


Fig. 9. Digital Signature involving Digital Certificates

The certification is so absolutely necessary in an infrastructure based on cryptography with public keys. But together with this come problems that must be solved, such as: certificate acquisition, its recognition, revocation, distribution and validation. The description of a standard digital signature that involves public certificates is presented in figure 9. In figure 9 the sender A hashes the original message M. The digest is sent into the smart card that process RSA with the private key stored in secure manner on the smart card. After the smart card processing the signature is attached to the original message using PKCS. The message with signature arrives to the receiver B. The receiver B gets the public key of A using a certification chain from the certificates stored on CA's servers. The public key of A and the signature are the inputs for RSA algorithm and it is obtained the digest H2. The original message is hashed with a dispersion function (MD5 or SHA-1) and it is obtained H1. If H1 is the same as H2 then the signature is valid.

#### 4. Conclusions

The study is important because it seems possible to have different certificates with the same signature value – last field in ASN.1 sequence of a certificate. More information about collisions is in [2]. They "announce a method for the construction of pairs of valid X.509 certificates in which the 'to be signed' parts form a collision for the MD5 hash function. As a result the issuer signatures in the certificates will be the same when the issuer uses MD5 as its hash function". Also, they show "that MD5 collisions can be crafted easily in such a way that the principles derlying the trust in Public Key Infrastructure are violated". There is in research how a tificate trust may be violated by the exploited of MD5 collision over digital certificates. That's why it is important to analysis at byte level of ASN.1 DER/BER encoding of the certificates and their implications in the lic key infrastructures.

#### References

- [1] B. Kaliski Jr. A Layman's Guide to a Subset of ASN.1, BER, and DER, RSA Publishing House, USA 1993.
- [2] A. Lenstra, X. Wang, and B. de Weger Colliding X.509 Certificates, http://www.win.tue.nl/~bdeweger/Collidi ngCertificates/
- [3] http://www.oid-info.com/ cgi-bin/display
- [4] C. Toma. Security in Software Distributed Platforms, AES Publishing House, Bucharest, 2008.
- [5] W. Rankl and Effing. Smart Card Handbook 3rd Edition, John Wiley & Sons Publishing House, USA 2004, reprinted 2007.



**Cristian TOMA** has graduated from the Faculty of Economic Cybernetics, Statistics and Informatics, Economic Informatics specialization, within Academy of Economic Studies Bucharest in 2003. He has graduated from the BRIE master program in 2005 and PhD stage in 2008. In present, he is assistant lecturer at Economic Informatics Department and he is involved in IT&C Security master program. His research areas are in: distributed and parallel computing, mobile applications, smart card programming, ebusiness and e-payment systems, network security, computer anti-viruses

and viruses, secure web technologies and computational cryptography. He is teaching assembly language, object oriented programming, data structures, distributed applications development and advanced programming languages in Economic Informatics Department and he has published 2 books and over 30 papers in indexed reviews and conferences proceedings.