ASN.1 specification for ETSI certificates and encoding performance study

Badis Hammi, Jean Philippe Monteuuis, Eduardo Salles Daniel, Houda Labiod
INFRES Laboratory, Telecom Paristech, France
badis.hammi, monteuuis, eduardo.sallesdaniel, houda.labiod@telecom-paristech.fr

***Abstract*—Cooperative Intelligent Transportation Systems (C-ITS) are gaining ground and are almost part of our everyday life. Within these environments, huge amounts of messages are exchanged. Besides, these messages should be secure in order to ensure users' privacy. Public Key Infrastructures (PKI) represent the most common security solution. Due to the vehicles speed, the communication with the PKI should be fully optimized. The European Telecommunications Standards Institute (ETSI) proposes a PKI architecture for C-ITS environments. However, unlike most of security standards as IEEE 1609.2, there is no Abstract Syntax Notation One (ASN.1) specification for the used certificates. For this reason, in this paper, we propose an ASN.1 definition for the ETSI certificate to help developers in its implementation. In addition, we provide an extensive comparative study of the different encoding schemes, applied to this proposal.**

***Index Terms*—C-ITS, ETSI, Certificate, Encoding schemes, ASN.1, Performance**

I. Introduction

During the last years, Collaborative Intelligent Transportation Systems (C-ITS) have gained momentum and will soon become essential to roads' users through the different applications they offer. therefore, such applications have evolved and become very greedy in terms of transmitted traffic and used bandwidth. Consequently, huge amounts of data and messages are exchanged continuously. Most of the exchanged messages contain critical data that should be confidential. Furthermore, numerous requested services require authentication to be accessed. To handle these security requirements, Public Key Infrastructures (PKIs) represent the most common solution.

However, using security within vehicular communications costs extra resources. Indeed, the use of wireless technologies for communication such as cellular or ITS-G5/802.11p has a strong impact on the network performance. For instance, messages' size represents a parameter that has to be seriously considered. The chosen encoding scheme has a strong impact on the size of the encoded data and on the encoding and decoding times.

In most of security standards such as IEEE1609.2 [8] and X509 [9], an Abstract Syntax Notation One (ASN.1) specification, a chosen encoding scheme of the described protocol and the used structures are provided. The goal of ASN.1 is the unicity of implementations through the restriction of interpretations and misunderstandings of the described standards and protocols. Nonetheless, multiple structures of European Telecommunications Standards Institute (ETSI) standards are not defined in ASN.1. ETSI certificate [7], a major component and player of security system is one of these non defined structures. Hence, in this paper we proposed an ASN.1 definition of the ETSI certificate. Beyond this proposal, we provided an extensive experimental study of the different encoding schemes applied to this use case. To the best of our knowledge, our work represent the first study that compares the performances of such a number of encoding schemes in such a context. Consequently, the contributions of this work are: (1) the restriction of wrong interpretations and misunderstandings of the ETSI certificate structure. Indeed, the definition of an ASN.1 structure for the ETSI certificate will help developers to follow the same constrictions and principles, accomplishing the ETSI standard policy; (2) providing a comparative study that allows the usage of the optimal encoding scheme, which allows best performances; and (3) making easier extensions on the certificate structure.

The rest of this paper is organized as follows: in Section II, ASN.1 language description is provided, as well as its different encoding schemes. Section III gives an overview of the related works. Section IV describes our ASN.1 proposal for the ETSI certificate and the followed experimental study that aims the comparison of encoding schemes. Section V discusses the obtained results. Finally, SectionVI concludes the paper and introduces our future works.

II. ASN.1 and data encoding

A. ASN.1 definition and utility

Abstract Syntax Notation One (ASN.1) is a standardized language used to describe and to specify, in an abstract way, data-structures, independently of the used platform and the implementation language. Actually, it exists numerous compilers that automatically translate ASN.1 structures into other language's syntax. Besides, in order to represent or store ASN.1 data-structures as hexadecimal or binary values, the use of ASN.1 encoding rules is needed. The latter are also used for the exchange of abstract data in network computing, where each ASN.1 structure becomes a protocol data unit (PDU) and is encoded in a determined number of bytes.

An ASN.1 data-structure can be composed of multiple ASN.1 structures. Thus, the resulting encoding will be the concatenation of the encodings realized on all the structures composing the root ASN.1 definition. The receiver is required to have the ASN.1 definition, in order to be able to decode the structure.

B. Encoding schemes

Numerous encoding rules for ASN.1 exist, e.g: BER, DER, PER, and XER. The latter aim to define a data representation, able of being transmitted as a transfer syntax. In other words, the encoding rules define how an ASN.1 structures are encoded into byte array forms, in order to be transmitted.

The encoding rules follow a defined parametrization called Type-Length-Value (TLV) as follows: (1) Type tag determines the ASN.1 element that is being encoded, i.e. SEQUENCE, CHOICE, INTEGER, etc. For example, the Type tag of SEQUENCE structure is '$30'_H$; (2) Length tag is build by computing the length of the Value field; (3) Value tag is the actual value of the specific structure or field. In the following, we describe the main ASN.1 encoding schemes.

1) Basic Encoding Rules (BER): The ASN.1 Basic Encoding Rules (BER) give one or multiple ways to represent any ASN.1 structure as an octet string. Its encoding rules follow the TLV policy and operate according to three methods:(1) Primitive with definite-length encoding; (2) Constructed with definite-length encoding; and (3) Constructed with indefinite-length encoding. Simple non-string types employ the primitive with definite-length method; structured types employ any constructed method; and simple string types employ any methods, but, depending on whether the length of the value is known or not; types derived by implicit tagging employ the method of the underlying type. Finally, types derived by explicit tagging employ the constructed methods [3].

2) Distinguished Encoding Rules (DER): The Distinguished encoding rules (DER) represent a subset of BER. These encoding rules are one of the most popular encoding formats, since they are used to encode and to store X.509 certificates in files [3]. Unlike BER, DER encodings have a unique encoding for each ASN.1 value. By the fact that DER is a subset of BER, one could BER-decode from a DER encoding but not vice versa. This is due to the fact that for the same ASN.1 value, BER have different encodings.

3) Packed Encoding Rules (PER) and Unaligned PER (UPER): Packed encoding rules, BASIC-PER or just PER encoding rules rise from BER encoding style, but more compacted. The aim of compacting the encodings occurred by the fact that BER encodings frequently have possible redundant octets. For instance, Boolean value has a fixed length. Thus, if we suppress the 'Length' of the TLV, one could still decode it correctly as we know the type of value we are dealing with, through the value of 'Type'.

When a structure is encoded, one obtains an array of bytes. However, not all the encoded structures always represent a complete byte array value. For example, a structure represented in 6 bits, one can represent it as a byte (8 bits), or just as a bit array of length 6. The Aligned PER version follows the octet-aligned-bit-fields policy and will insert padding bits to the unfulfilled octets, while in the Unaligned PER (UPER) version, no padding bits are inserted [4].

4) Octet Encoding Rules (OER): OER represents a mix of BER and PER. More precisely, OER follows the octet-aligned-bit-fields policy, as Aligned PER. However, it removes 'Type' and 'Length' tags [6].

5) Canonical Encoding Rules (CER): As DER, CER encoding rules, are a subset of BER. These encoding rules produce an unequivocal transfer syntax for data structures described by ASN.1. Whereas BER gives choices, such as how data values may be encoded. CER -as DER- selects just one encoding from those allowed by the basic encoding rules. Eliminating, thus, the rest of the options [3]. CER and DER differ in the set of restrictions that they place on the encoder. The basic difference between them, is that DER uses definitive length form and CER uses indefinite length form.

6) XML Encoding Rules (XER): XML encoding rules (XER) represent verbose representations of ASN.1 structures. These encoding rules were developed as a link between XML and ASN.1. They are specially useful for interactions with a system configured just to understand XML. XER encoding rules attempt to bridge the gap by providing a textual encoding of defined data structures using ASN.1 notation. Nevertheless, it does not provide any user control over the produced style of XML [5]. In the latter, the 'Type' tag is represented as the name of the elements (SEQUENCE, CHOICE,...).

EXTENDED-XER (E-XER) is a variation of BASIC-XER encodings. They increase verbosity by specifying a set of XER encoding instructions associated with ASN.1 types. In absence of XER encoding instructions, an EXTENDED-XER encoding differs from a BASIC-XER encoding only because it provides more encoders options.

III. Related works

Several studies were been provided in order to compare performances of encoding schemes. However, few ones have targeted the ITS and vehicular environments.

Darren *et al.*[11] and authors in [12], have realized a performance comparison between ASN.1 and XML. The conclusion is that XML is not an effective solution for providing a human readable and easy to parse data representation at the cost of bigger encoding length. Therefore, this cannot be considered as a practical option for ITS communications where the length of encoded data needs to be as small as possible.

Bittl *et al.*[13] proposed a performance comparison of encoding schemes for C2X communications based on ETSI standards. In their experiment, the computation time, the memory footprint and the encoded data length, were the three considered parameters of the performance evaluation for each encoding scheme. The evaluated data structure was ETSI Secured Message [7], considering the following encoding schemes: binary, Protocol Buffers (Protobuff) [2], Efficient XML Interchange (EXI) [1] and ASN.1. The authors concluded that ASN.1 encoding outperformed Protobuff and EXI encodings but not binary encoding. Besides, ASN.1 encoding scheme has better results for data length. However, this paper does not provide any details about programming language and especially on how the certificate and secured message have been defined for each syntax, which brings questions about structure optimizations. Furthermore, they used only

Unaligned PER ASN.1 encoding scheme, which is restrictive considering the number of existing ASN.1 encoding schemes, especially, because UPER is not the most optimal one.

IV. Proposal of ASN.1 structure for ETSI certificate

In this section we present and detail our ASN.1 proposal for the ETSI certificate and the performance study that we conducted in order to compare the different encoding schemes.

A. ASN.1 certificate

ETSI certificate is described by the standard ETSI TS 103097 [7]. It is composed of six principal structures :

- Version: represents the version of the certificate, currently version 2.
- SignerInfo: contains relevant information about the authority signing the certificate. This field is needed in order to identify the signer when checking the signature of the certificate.
- SubjectInfo: specifies the type of the subject owning the certificate, e.g. ITS station (ITSS) or PKI authority and its name (subject name).
- SubjectAttribute: comprises data used by security functions and includes:
  - The verification public key used to generate Elliptic Curve Digital Signature Algorithm (ECDSA) signatures.
  - The encryption public key used for the Elliptic Curve Integrated Encryption Scheme (ECIES) encryption.
  - The reconstruction value containing an ECCPoint (in the case of using implicit certificates).
  - The assurance level associated to the certificate.
  - The list of Specific Service Permissions (SSPs) associated to the certificate
- ValidityRestriction: specifies the validity of the certificate. Two types of validity restriction are defined in this field: (1) time restrictions and (2) region restrictions.
- Signature: covers all the mentioned fields and performed using the private verification key of the authority described in SignerInfo.

The different structures of ETSI TS 103097 standard are described by the meaning of a syntax derived from IETF RFC 2246 [10] and from IEEE 1609.2-2012 [8]. However, due to this syntax ambiguity, the implementation of these structures can lead to numerous interpretations. In other words, multiple implementations can be derived from this description (even those that do not respect the standard). Our motivation behind this work is the proposal of an ASN.1 structure that allows a correct implementation of the certificate.

Listing 1 contains our ASN.1 proposal for the ETSI certificate. This proposal accurately reflects the one described by ETSI TS 103097 except for few details. These changes were provided in order to secure the generation and use of the certificate. Indeed, due to the flexibility of these structures, they are subject to erroneous generation or to manipulation for malicious purposes. These changes are as follows:
(1) According to the ETSI TS 103097 standard [7] the signerInfoType can be one of the following: (1) self; (2) certificate digest with SHA256; (3) certificate digest with other algorithms; (4) certificate; or (5) certificate chain. However, in the section dedicated to this structure, it is clearly explained that this type have to be only one of the three following : (1) self; (2) certificate digest with SHA256; or (3) certificate digest with an other algorithm. In order to avoid that users choose wrongly unauthorized types, in our proposal, we restricted the choice of the signerInfoType to only authorized types.
(2) [7] describes SubjectAttributes as a vector of variable length which can contain different types of SubjectAttributes. However, it is clearly explained further, that this structure does not support two SubjectAttributes of the same type. Furthermore, a verification key and an assurance level SubjectAttribute are mandatory. In order to be compliant with these conditions and to avoid wrong SignerInfo generations, a sequence of multiple SubjectAttributes is defined, including the verification key and assurance level SubjectAttributes as mandatory and the rest as optional.
(3) In the last structure, according to the type of the certificate, an ITS Application IDentifier List (ITS AID List) or an ITS AID Service Specific Permissions (SSP) List is defined, but never both. Even though, as it is defined in the standard, it is possible to generate both of them. To remedy this problem, we define, in the list of SubjectAttributes an attribute called ITSList which consists of a choice between the two different list structures.
(4) ValidityRestrictions field, should contain a time ValidityRestriction and can contain a region ValidityRestriction. The certificate can comprise multiple ValidityRestrictions in a vector of variable length. Nonetheless, it could not comprise two time restrictions. But, the structure proposed by the standard does not fullfill this condition. To remedy this issue, we define the ValidityRestrictions field as a vector of variable length which can contain only one time ValidityRestriction. It is a choice between all the restrictions of type time. In addition, the vector can comprise unlimited number of region restrictions.
(5) In the last structure, a region Validity Restriction could be of different types, such as: circular, rectangular or polygonal. The rectangular region is defined as a composition of multiple rectangles, at least one and maximum of six. However, the proposed structure does not guarantee this condition. Our solution remedy this problem, by requiring at least one rectangle as well as limiting the maximum number of rectangles to six.
(6) According to [7], different compression modes for Elliptic Curve Cryptography (ECC) Points are used. However, the following restrictions have to be followed: For the Verification Key and the Encryption key, $x - coordinate - only$ type should not be used; In Signature field, the r field is an ECC Point and does not support $uncompressed$ type; finally, the reconstructionValue has no constraints in terms of type of compression. To avoid confusion and wrong generations, in our solution, we propose the usage of three ECC Point

structures, one for each described type.

Listing 1: ASN1 definition of the ETSI certificate

```
Certificate103097 DEFINITIONS AUTOMATIC TAGS
    ::= BEGIN
IMPORTS
 Uint8, Uint16, HashedId8, Time32,
 TwoDLocation, CircularRegion, Duration,
 RectangularRegion, HashAlgorithm,
 SubjectAssurance, SymmAlgorithm
FROM IEEE1609dot2BaseTypes {iso(1)
identified-organization(3) ieee(111)
standards-association-numbered-series-
standards(2) wave-stds (1609)dot2(2)
base(1) base-types(2)}
Certificate ::= SEQUENCE {
 version Version,
 signerInfo SignerInfo,
 subjectInfo SubjectInfo,
 subjectAttributes SubjectAttributes,
 validityRestrictions ValidityRestrictions,
 signature Signature
}
Version ::= Uint8
 SignerInfo ::= CHOICE {
 self NULL,
 certificateDigestSHA256
     CertificateDigestSHA256,
 certificateDigestOtherAlgorithm
     CertificateDigestOtherAlgorithm
}
CertificateDigestSHA256 ::= HashedId8
CertificateDigestOtherAlgorithm ::= SEQUENCE {
 algorithm HashAlgorithm,
 digest HashedId8
}
SubjectInfo ::= SEQUENCE {
 subjectType SubjectType,
 subjectName UTF8String (SIZE(0..32))
}
SubjectType ::= ENUMERATED {
 enrollment-credential(0),
 authorization-ticket(1),
 authorization-authority(2),
 enrollment-authority(3),
 root-ca(4),
 crl-signer(5)
}
SubjectAttributes ::= SEQUENCE {
 verificationKey VerificationKey,
 encryptionKey EncryptionKey OPTIONAL,
 assuranceLevel SubjectAssurance,
 reconstructionValue RecontructionEccPoint
     OPTIONAL,
 itsList ITSList,
 ...
}
PublicKey ::= SEQUENCE {
 algorithm PublicKeyAlgorithm,
 key EccPoint
}
PublicKeyAlgorithm ::= ENUMERATED {
 ecdsa-nistp256-with-sha256(0),
 ecies-nistp256(1),
 ...
}
EccPoint ::= CHOICE {
 compressed-y-0 OCTET STRING (SIZE (32)),
 compressed-y-1 OCTET STRING (SIZE (32)),
 uncompressed SEQUENCE {
    x OCTET STRING (SIZE (32)),
    y OCTET STRING (SIZE (32))
  }
}
RecontructionEccPoint ::= CHOICE {
 x-coordinate-only OCTET STRING (SIZE (32)),
 compressed-y-0 OCTET STRING (SIZE (32)),
 compressed-y-1 OCTET STRING (SIZE (32)),
 uncompressed SEQUENCE {
    x OCTET STRING (SIZE (32)),
    y OCTET STRING (SIZE (32))
  }
}
VerificationKey ::= PublicKey
EncryptionKey ::= SEQUENCE {
 supported-symm-alg SymmAlgorithm,
 public-key PublicKey
}
ITSList ::= CHOICE {
 its-aid-list ITS-AID-LIST,
 its-aid-ssp-list ITS-AID-SSP-LIST
}
ITS-AID-LIST ::= SEQUENCE OF ITS-AID
ITS-AID ::= INTEGER
ITS-AID-SSP-LIST ::= SEQUENCE OF ItsAidSsp
ItsAidSsp ::= SEQUENCE {
 its-aid ITS-AID,
 service-specific-permissions OCTET STRING
     (SIZE(0..31))
}
ValidityRestrictions ::= SEQUENCE {
 timeOfValidation ValidityRestrictionTime,
 region ValidityRestrictionRegion OPTIONAL,
 ...
}
ValidityRestrictionTime ::= CHOICE {
 timeEnd Time32,
 timeStartAndEnd TimeStartAndEnd,
 timeStartAndDuration TimeStartAndDuration
}
TimeStartAndEnd ::= SEQUENCE {
 startValidity Time32,
 endValidity Time32
}
TimeStartAndDuration ::= SEQUENCE {
 startValidity Time32,
 duration Duration
}
ValidityRestrictionRegion ::= CHOICE {
 circularRegion CircularRegion,
 rectangularRegion RectangularRegions,
 polygonalRegion PolygonalRegion,
 identifiedRegion IdentifiedRegion,
 ...
}
RectangularRegions ::= SEQUENCE SIZE(1..6) OF
    RectangularRegion
PolygonalRegion ::= SEQUENCE SIZE(3..12) OF
    TwoDLocation
IdentifiedRegion ::= SEQUENCE {
 regionDictionary RegionDictionary,
 regionIdentifier Uint16,
```

```
 localRegion INTEGER
}
RegionDictionary ::= ENUMERATED {
 iso-3166-1(0),
 un-stats(1)
}
Signature ::= EcdsaP256Signature
EcdsaP256Signature ::= SEQUENCE {
 r SignatureEccPoint,
 s OCTET STRING (SIZE (32))
}
SignatureEccPoint ::= CHOICE {
 x-coordinate-only OCTET STRING (SIZE (32)),
 compressed-y-0 OCTET STRING (SIZE (32)),
 compressed-y-1 OCTET STRING (SIZE (32))
}
END
```

B. Performance study

1) Overall context, experimental framework and scenarios: We consider the case of a C-ITS environment, in which two ITSS communicate. Within a communication, a message is encoded by the first station, transmitted and finally decoded by the second one. For authentication purposes, the message contains the sender's certificate. Knowing that certain messages are sent extensively and periodically over a wireless network characterized by its components' high speed, it is very important to choose the optimal data encoding scheme for transmission. To that aim, we realized an extensive experimental campaign that study and compare the different encoding schemes. The latter are : Binary in BigEndian form, PER, UPER, DER, BER, OER, COER, XER, CXER and EXER. To the best of our knowledge, this work is the first that aims to compare all these encoding schemes in a cooperative ITS use case. This study concerns only the ETSI certificate. It considers only the encoding of certificate, without taking into account the structure of the message (secured messages) including this certificate.

For our experimentations, we used two computers; sender and receiver. Both have the same technical features, namely Intel® Xeon® CPU E5-1607 v3 @ 3.10GHz (quad-core) with 8 Giga bytes of RAM. The implementation of the described ASN.1 structure was realized using OSS Nokalva compiler[1]. The latter generates also the Java code of the conceived certificate and provides Java libraries that support all the ASN.1 encoding rules. Consequently, we can also encode and decode the created Java objects. For detailed scenarios, as described by Algorithm 1, the sender generates a random certificate. The randomization covers all the fields of the certificate. After, the generated certificate is encoded sequentially with the mentioned encodings schemes(Binary, PER, UPER, DER, BER, OER, COER, XER, CXER and EXER). For each encoding, the size of the encoded certificate as well as the time of the encoding are stored. In order to get accurate results and deeply feature encodings from time and size perspectives, we realized numerous experimentations by varying, the number of generated certificates, defined by the X parameter. Thus, X takes the following values: 10, 10^2, 10^3, 10^4, 10^5 and 10^6. Consequently, we realized six experimentations, each realizes encoding and decoding following 10 encoding schemes.

From the receiver side, as described by Algorithm 2, the station decodes each received certificate. Then, For each encoding scheme, it stores the decoding time.

[1]www.oss.com/asn1/products/asn1-java/asn1-java.html

Algorithm 1: Sender

Declaration:
X: Integer ▷ Nb of wanted certificates
Enc: Tab[10] String ▷ Available encoding schemes
$EncT$: Tab[10] Files ▷ Files storing encoding times
$EncS$: Tab[10] Files ▷ Files storing encoding sizes
$Cert$: Obj ▷ Certificate

1: **procedure** ENCODEANDSENDCERIFICATE
2: **for** i *in* 1 *:* X **do**
3: $Cert \leftarrow$ GENERATERANDOMCERTIFICATE()
4: **for** j *in* 1 *:* $Length(Enc)$ **do**
5: ENCODE(ENC[J], CERT)
6: SAVEENCODINGTIME($EncT[j]$)
7: SAVEENCODINGSIZE($EncS[j]$)
8: **end procedure**

Algorithm 2: Receiver

Declaration:
$DecT$: Tab[10] Files ▷ Files storing decoding times
$Cert$: Obj ▷ Certificate

1: **procedure** RECEIVEANDDECODECERIFICATE
2: $Cert \leftarrow$ RECEIVECERTIFICATE()
3: DECODE(CERT)
4: SAVEDECODINGTIME($DecT[cert.encoding]$)
5: **end procedure**

We are aware about the difference of performance between computers such as those used in our experimentations and real On Board Units (OBU). We are also aware about the fact that the obtained results depends on the used language (Java) and are different when using other languages. However the goal of our study is the comparison of the different encoding schemes. Consequently, the comparison is fair since all encodings and decodings are realized on the same basis, language and material.

V. EXPERIMENTAL RESULTS AND DISCUSSION

In this section we present the different results that we obtained through our evaluations. The latter covers: (1) encoding time, (2) decoding time, (3) certificate' size and (4) encoding and decoding speeds. For each metric, we compare the obtained results of ASN.1 encodings with Binary, the actual used encoding. Knowing that our material is powerful compared to the one used on board of vehicles, we consider an accuracy of at least 10^{-8}, in order to show differences that can occur in a limited material.

Encoding	T. enc. Average (ms)	T. enc. Var.	T. enc. SD	T. dec. Average (ms)	T. dec. Var.	T. dec SD	Size average (bytes)	Size Var.	Size SD
Binary	0.01833026	0.0006636195	0.02576081	0.04438053	0.02209007	0.1486273	200.4964	479.9544	21.90786
UPER	0.01918138	0.0002978974	0.01725971	0.02672955	0.0004372456	0.02091042	187.3701	468.7203	21.64995
PER	0.02144488	0.0005857316	0.02420189	0.03450128	0.003040983	0.05514511	192.3701	468.7203	21.64995
DER	0.02054847	0.001949473	0.04415283	0.02232729	0.002396427	0.04895331	251.0994	632.6233	25.152
BER	0.02057498	0.001595099	0.03993869	0.03037089	0.0020046642	0.04477323	251.0994	632.6233	25.152
OER	0.01834905	0.0005165277	0.02272725	0.02283728	0.0007253007	0.02693141	198.3701	468.7203	21.64995
COER	0.01405544	0.0004534607	0.02129462	0.01549953	0.0005262207	0.0229395	198.3701	468.7203	21.64995
XER	0.03533994	0.001298747	0.03603814	0.09933497	0.00264926	0.05147096	1924.137	5959.503	77.19782
CXER	0.02967197	0.001009983	0.03178023	0.08626314	0.002176599	0.04665404	1496.422	2496.244	49.96243
EXER	0.04752311	0.001929555	0.0439267	0.18197	0.005501827	0.0741743	1880.13	5251.959	72.4704

TABLE I: Statistics (Average, Variance and Standard Deviation) of the obtained results with 10^6 certificates

A. Encoding Time

Figure 1.a describes the evolution of the encoding times sums[2], over the different realized experimentations, for each different encoding scheme. We can note that, (1) for number of generated certificates taken between 10 and 10^3, the sum of the different encoding times is very close. (2) From 10^3 to 10^5 the differences between the encoding times begin to be shaped. (3) Finally, from 10^5 and up, we obtain two groups, XML encodings group and the rest. The second group contains close values following the next ranking (the first has the shortest encoding time, thus the best one): (1) COER, (2) Binary, (3) OER, (4) UPER, (5) DER, (6) BER, (7) PER, (8) CXER, (9) XER, (10) EXER.

For more precision on the obtained results, Table I describes the average, the variance and the standard deviation (SD) of encoding time, decoding time and size of the generated certificates for all the used encoding schemes. The first 3 columns describe statistics on the obtained encoding time as follows: (1) COER have 0.01405544 milliseconds (ms) of average, 0.0004534607 of variance and 0.02129462 ms of SD. (2) Binary, slower than COER with 0.00427482 ms, thus having an average of 0.01833026 ms, 0.0006636195 of variance which represent the biggest variance after XML encodings (XER, CXER and EXER) and DER, and finally 0.02576081 ms of SD. (3) OER with 0.01834905 ms, thus, slower with only 0.00001879 ms than Binary. (4) UPER with 0.01918138 ms of average encoding time, thus, slower than Binary with 0.00085112 ms. A variance of 0.0002978974, and an SD of 0.01725971 ms. UPER has the smallest obtained variance and SD values. Consequently, we can conclude that UPER is the most stable for encoding. (5) DER with 0.02054847 ms of average (0.00221821 ms slower than Binary), 0.001949473 of variance and 0.04415283 ms of SD. The latter are the biggest obtained values, leading DER to be the most unstable encoding scheme. (6) BER, very close to DER with 0.02057498 ms of average, 0.00224472 ms slower than Binary. (7) PER, with 0.02144488 ms of average time, thus 0.00311462 ms slower than Binary. (8) CXER, having 0.02967197 ms of average time, thus, 0.01134171 ms slower than Binary. (9) XER, slower than Binary with 0.01700968 ms by having 0.03533994 ms of average. (10) Finally, EXER, with 0.04752311 ms of average time (0.02919285 ms slower than Binary).

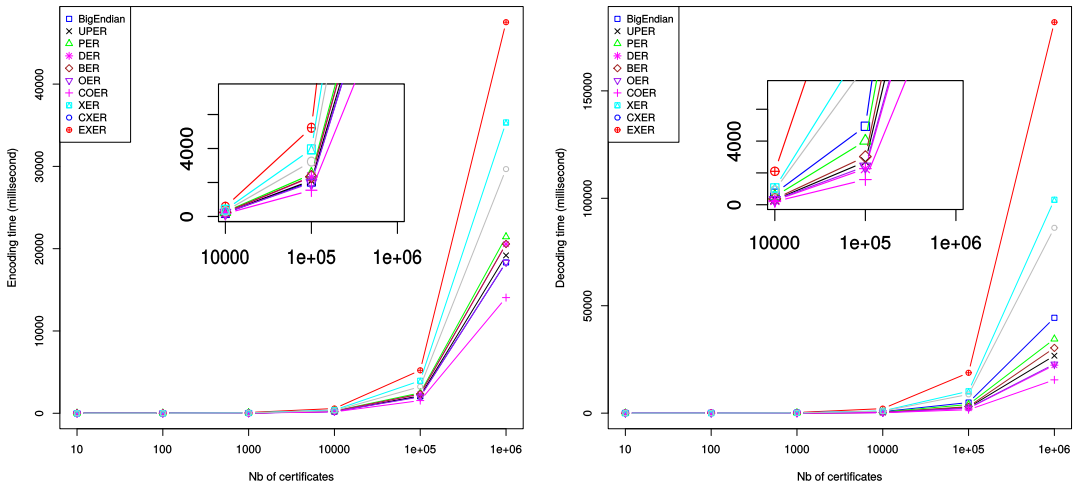
Even if the interval composed by the SD $[Average - SD, Average + SD]$ is very narrow, we can not have an accurate idea of the obtained encoding times values. Thus, Figure 2.a describes the boxplot realized on the encoding time values of the experimentation involving the generation of 10^6 certificates. For clarity purposes, we focus only on the small interval $[0, 0.1]$ ms, which includes more than 75% of all values. A boxplot is a statistical graphical representation, where, (1) the minimum, (2) first quartile, (3) median, (4) third quartile, and (5) the maximum of a statistical population. Thus Figure 2.a highlights where most of encoding times values are concentrated.

B. Decoding Time

Figure 1.b describes the evolution of the decoding times sums, over the different realized experimentations, for each different encoding scheme. We can note that, (1) when the number of generated certificates is between 10 and 10^3, the sum of the different decoding times is almost invariant. (2) From 10^3 to 10^5 the differences between the decoding times begin to be shaped. (3) Finally, from 10^5 an up, we obtain, contrarily to encoding times case, a clear ranking of the different sums.

The $5^{th}, 6^{th}$ and 7^{th} Columns of Table I describe the average, variance and SD of the obtained decoding times. One can note that the obtained ranking is different from the ranking of encoding times: (1) COER keeps the first place of the ranking with an average time of 0.01549953 ms, which means 0.028881 ms shorter than Binary. (2) DER, with 0.02232729 ms of average time (0.02205324 ms shorter than Binary). (3) OER, having 0.02283728 ms, thus 0.02154325 ms less than Binary. (4) UPER, having 0.02672955 ms of average (0.01765098 ms less than Binary). It has a variance of 0.0004372456 and 0.02091042 of SD. As in the case of encoding, variance and SD values are the smallest obtained,

[2]Since the messages are sent continuously by an OBU, we believe that using the sum for comparison is more explicit for highlighting times' differences than the average



(a) (b)

Fig. 1: Evolution of the sum of: (a) Encoding times; (b) Decoding times

witnessing of UPER's stability. (5) BER, with 0.03037089 ms of average time (0.01400964 ms less than Binary). (6) PER, having 0.03450128 ms of average time. Which means 0.00987925 less than Binary. (7) Binary, with 0.04438053 ms of average time, having a variance of 0.02209007 and an SD of 0.1486273 ms. We can note that this decoding time is worst than all ASN.1 schemes, except XML ones. (8) CXER, with 0.08626314 ms of average time, which means 0.04188261 ms more than Binary. (9) XER, having 0.09933497 ms of average (0.05495444 ms more than Binary). Finally (10) EXER with 0.18197 ms of average time. Thus, 0.1375895 ms more than the reference encoding.

Figure 2.b describes the boxplot realized on the decoding time values of the experimentation involving the generation of 10^6 certificates. For clarity purposes, we focus only on the small interval $[0, 0.1]$ ms, which includes more than 75% of the values.

C. Certificate's size

The last 3 columns of Table I describe statistics of the obtained size values. Figure 2.c describes the boxplot realized on the size values of the experimentation involving the generation of 1 million certificates. From these results we can conclude the following ranking: (1) UPER, with 187.3701 bytes of average size, thus, smaller by 13.1263 bytes from Binary. (2) PER, having 192.3701 bytes of average size and thus 8.1263 bytes smaller than Binary. (3) COER and OER with 198.3701 bytes of average size, 2.1263 bytes smaller than the reference. (5) Binary with 200.4964 bytes of average size. (6) DER and BER, having 251.0994 bytes of average size, thus, 50.603 bytes more than the average size of Binary. (8) and very far, CXER with 1496.422 bytes of average size, having 1295.926 bytes more than the average size of Binary. (9) EXER with 1880.13 bytes. Consequently, it has 1679.634 bytes more than the reference. Finally (10) XER with 1924.137 bytes, which means 1723.641 bytes more than the reference.

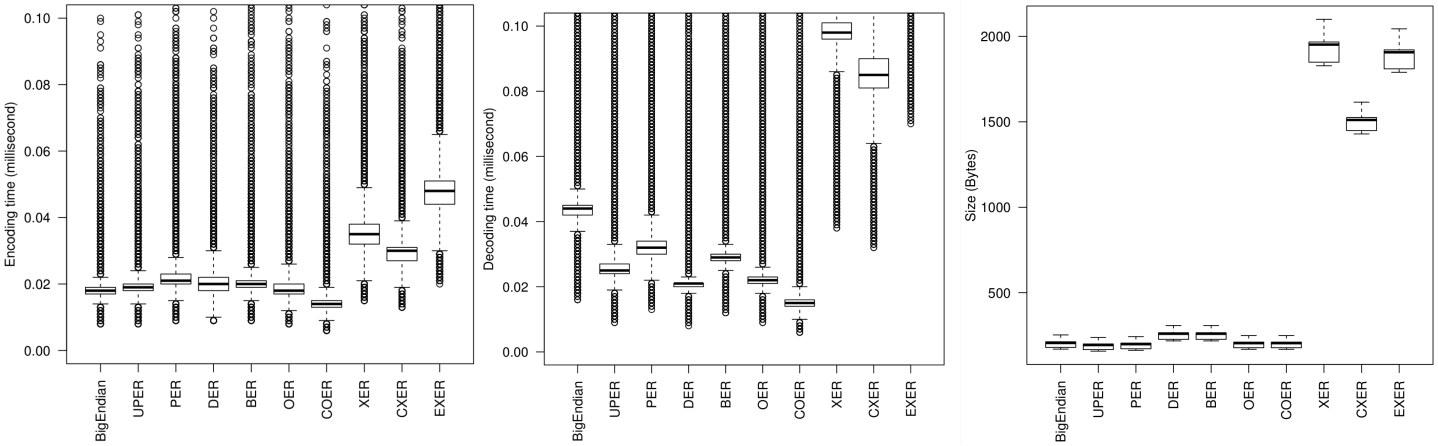
D. Encoding and decoding speeds

In this section, we discuss encoding and decoding speeds of the different encoding schemes. A speed is calculated according to the average values using the equation $Speed = \frac{\delta size}{\delta time}$.

If we sort these speeds in order to get a ranking of how fast are the different schemes, we obtain the following ranking. For encoding: (1) the fastest encoding scheme is XER, with 54446.53 Bytes/millisecond (B/ms); (2) CXER having a speed of 50432.18 B/ms; (3) EXER, with 39562.44 B/ms; (4) COER having a speed of 14113.4 B/ms; (5) DER with 12219.86 B/ms. (6) BER with 12204.11 B/ms; (7) Binary with 10938 B/ms; (8) OER having 10810.92 B/ms; (9) UPER with 9768.333 B/ms and finally (10) PER with 8970.444 B/ms.
For decoding: (1) the fastest decoding scheme is always XER, with 19370.19 B/ms; (2) CXER with 17347.18 B/ms; (3) COER having a speed of 12798.46 B/ms; (4) DER with 11246.3 B/ms; (5) EXER with 10332.09 B/ms; (6) OER having 8686.24 B/ms; (7) BER having a speed of 8267.766 B/ms; (8) UPER having 7009.849 B/ms of speed; (9) PER with 5575.738 B/ms, and finally (10) Binary having a speed of 4517.666 B/ms

To summarize, according to needs, the choice of the encoding scheme could be different. Indeed, COER realizes the best encoding and decoding times, UPER realizes enormous size savings. We can also note that XML encodings are very fast comparing to other encodings but completely not adapted to this use case.

We recall that the experimentations were realized on powerful computers compared to those used on board of vehicles.



(a) (b) (c)

Fig. 2: Boxplot of: (a) Encoding time; (b) Decoding Time, (c) Size

Thus, in real OBUs, values' differences are more important than those we obtained. However, the comparison still fair, since all encodings and decodings are realized on the same basis, language and material.

VI. Conclusion and future works

For security purposes, the main transmitted ETSI ITS messages include sender's certificate. Generally, security standards have their own ASN.1 specifications, e.g. IEEE 1609.2 certificate, in order to make easy their implementation and avoid their wrong understanding. However, currently, there is no existing ASN.1 definition for ETSI certificates. For this reason, in this paper we proposed an ASN.1 definition for the ETSI certificate as described in the standard ETSI TS 103097 [7]. This proposal accurately reflects the one described by the standard and allows a secure generation of the certificate. Indeed, due to the ambiguity of the used description language, the described structures are subject to erroneous generation or manipulation.

Beyond this proposal, we provided an extensive study of the different encoding schemes. To the best of our knowledge, it represents the first study that compares the performances of numerous ASN.1 encoding schemes and compare them to Binary encoding scheme in a C-ITS context. This study proves clearly that some ASN.1 encoding schemes are more performant than Binary encoding scheme and their application enhance the networks' performances.

For future works, at a short term, we plan to submit our ASN.1 definition to ETSI in order to be used for the standard description. We also intend to specify an ASN.1 structure for the ETSI secured message and to provide a performance study on which encoding could be the most suitable for it. At a long term perspective, we plan to provide an experimental implementation of the described structures, in a real C-ITS environment and to provide an extensive analysis of their usage.

Acknowledgment

This work is part of SCOOP@F project. SCOOP@F is a nationwide deployment project, led by the french government and involving numerous industrial partners (e.g. French car manufacturers: Renaults, PSA), road management companies (e.g. SANEF) and research institutions (e.g. Telecom ParisTech, Cerema). This is the largest experiment in Europe, with the deployment of approximately 3,000 intelligent vehicles over 2,000 km of connected roads.

References

[1] Efficient XML Interchange (EXI) Format 1.0 (Second Edition). *W3C Recommendation*, February 2014.

[2] Encoding - Protocol Buffers - Google Developers. *Google Recommendation*, September 2014.

[3] ITU-T X.690. ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER). *ITU-T Recommendation X.690*, August 2015.

[4] ITU-T X.691 (08/2015) ASN.1 encoding rules: Specification of Packed Encoding Rules (PER). *ITU-T Recommendation X.691*, August 2015.

[5] ITU-T X.693 (08/2015) ASN.1 encoding rules: XML Encoding Rules (XER). *ITU-T Recommendation X.693*, August 2015.

[6] ITU-T X.696 (08/2015) ASN.1 encoding rules: Specification of Octet Encoding Rules (OER). *ITU-T Recommendation X.696*, August 2015.

[7] ETSI TS 103 097 V1.2.1: Intelligent Transport Systems (ITS), Security header and certificate formats. June 2015.

[8] Intelligent Transportation Systems Committee et al. Ieee standard for wireless access in vehicular environments-security services for applications and management messages. *IEEE Vehicular Technology Society Standard*, 1609.2, January 2016.

[9] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. *IETF, May*, 2008.

[10] T. Dierks and C. Allen. The tls protocol version 1.0. *IETF, January*, 1999.

[11] Darren P Mundy, David Chadwick, and Andrew Smith. Comparing the performance of abstract syntax notation one (ASN.1) vs eXtensible Markup Language (XML). In *Proceedings Terena Networking Conference*, 2003.

[12] Inc OSS Nokalva. Alternative Binary Representations of the XML Information Set based on ASN.1. August 2013.

[13] M. Spähn S. Bittl, A. A. Gonzalez and W. Heidrich. Performance Comparison of Data Serialization Schemes for ETSI ITS Car-to-X Communication Systems. 2015.