Performance Analysis of Existing 1609.2 Encodings v ASN.1

Article *in* SAE International Journal of Passenger Cars - Electronic and Electrical Systems · May 2015

DOI: 10.4271/2015-01-0288

2 authors:

Virendra Kumar
Security Innovation, Inc.
15 PUBLICATIONS **378** CITATIONS
SEE PROFILE

William Whyte
OnBoard Security
47 PUBLICATIONS **680** CITATIONS
SEE PROFILE

Performance Analysis of Existing 1609.2 Encodings v ASN.1

Author, co-author (Do NOT enter this information. It will be pulled from participant tab in MyTechZone)

Affiliation (Do NOT enter this information. It will be pulled from participant tab in MyTechZone)

Abstract

IEEE Standard 1609.2-2013, Security Services for Applications and Management Messages for Wireless Access in Vehicular Environments (WAVE), specifies its data structures and encoding using a proprietary language based on that used in the Internet Engineering Task Force (IETF)'s Transport Layer Security (TLS) specification. This approach is believed to allow fast encoding and decoding, but is non-standard, is not proved to be complete, lacks automatic tools for generation of codecs, and is difficult to extend. For these reasons, the 1609 Working Group approved the use of Abstract Syntax Notation 1 (ASN.1) for future versions of 1609.2, so long as ASN.1 did not significantly degrade performance.

This paper is the first publication of the results of a performance analysis carried out to determine whether ASN.1-based encoding was in fact acceptable. We consider the issues in designing a security protocol, such as supporting one-pass processing, as well as issues that affect processing time and memory requirements, such as byte-alignment. We show that a design that is optimized for encode/decode time is not necessarily optimized for other metrics, such as ability to support one-pass processing or size of encoded messages. We present results on encoding size and encode/decode time for four alternatives: (a) ASN.1 with Packed Encoding Rules (PER) Unaligned, with a schema optimized for processing time; (b, c) ASN, with a schema, optimized for the requirements of a security protocol, using PER Unaligned and the recently-published Octet Encoding Rules (OER), respectively; and (d) the current 1609.2 protocol.

Our results show that ASN.1 can be used without a significant performance hit, and in fact that the OER may allow faster operations than current 1609.2 implementations. These results in this paper were a significant contributing factor to the 1609 Working Group's October 2014 decision to adopt ASN.1 with OER going forward.

Introduction

The IEEE P1609 working group has been developing standards for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I)[1] wireless communications [1] [2] [3]. These communications enable a range of applications that rely on communications between road users, including vehicle safety, public service, commercial fleet management, tolling, and other operations. The United States Department of Transportation (USDOT) has made a series of announcements, starting in February 2014 [7], that it intends to seek a mandate for the inclusion of V2V communications capabilities in all light vehicles from some future date, and has indicated [8] that the specific technology mandated is expected to be the technology specified in the 1609 series and related standards [4] [5] [6].

For applications that are safety-critical, and also otherwise, it is crucial that the messages be protected from attacks such as eavesdropping, spoofing, alteration, and replay. More importantly, as these technologies will be deployed in personal vehicles, it is very important that Personally Identifiable Information (PII) be not leaked to unauthorized parties. The ad-hoc nature of WAVE coupled with time-critical safety (and several other) applications present challenging constraints. The processing and bandwidth overhead due to security must be kept to a minimum, to improve responsiveness and decrease the likelihood of packet loss. Due to the very wide audience potentially consisting of several hundreds of millions of vehicles in North America, the mechanism used to authenticate messages must be as flexible and scalable as possible, and must accommodate the smooth removal of compromised WAVE devices from the system. On top of all that, the applications running on personal vehicles must respect the owner's privacy as far as technically and administratively feasible.

For these reasons, although there are existing standards for secure communications over the Internet [9] [10], the 1609 working group has developed its own WAVE-specific secure communications protocols. These protocols are specified in IEEE Std 1609.2-2013 [1] (hereafter referred to simply as 1609.2). 1609.2 specifies certain data structures and their corresponding encoding mechanisms, which are used to transfer information relevant to communications security services between two or more devices. The data structures for 1609.2 are specified in a proprietary format based on that used in Transport Layer Security (TLS) defined by the Internet Engineering Task Force (IETF). (See in particular the base spec [15]). Although this format has been used in many implementations and is reasonably well understood, it has a number of drawbacks: it is not a formal standardized language; it has no proofs of completeness or correctness; there are only few automated tools available to compile it; and it mixes identification of the information elements with a definition of their encoding.

To overcome these shortcomings, the IEEE 1609 working group approved a research project to investigate modifications to the standard so that the data structures are specified in Abstract Syntax

[1] V2V and V2I are together referred as V2X.

Notation 1 (ASN.1) [11]. ASN.1 addresses all of the shortcomings identified above: it is a formal standardized language with a proof of correctness (formally, it is a Backus-Naur form); there are many automated tools available; and the data structures are specified independently of their encoding, with a number of standardized encoding rules that can be chosen from. Additionally, the use of ASN.1 is consistent with guidance from the US Department of Transportation (USDOT) and the European Telecommunications Standards Institute (ETSI), which is standardizing V2V communications security in Europe. They suggest that standards that receive their financial support should be specified in ASN.1 where possible.

The group approved a research project rather than deciding to move to ASN.1 immediately because the 1609.2 specification is designed for use in constrained devices, where it is important to avoid excessive processing and memory allocation, and in situations of constrained bandwidth. Before making a final decision to move to ASN.1, it was necessary to demonstrate that this does not carry any significant performance costs. This is particularly the case for decoding, as devices will receive many more messages than they transmit.

This paper presents the results of that research project. We demonstrate that both the packet size and the decoding speed impacts of using ASN.1 are minor and acceptable. Based on the results presented in this paper, the 1609 working group voted in October 2014 to use ASN.1 for future versions of 1609.2.

Background

In V2X communication, especially involving safety-of-life applications, low processing latency is of critical importance. Devices may typically receive up to 1000 incoming messages per second [32], and driver warning applications have less than $1/10^{th}$ of a second to decide whether or not to act on a particular message. This means that signature verification done in software that takes around 50 milliseconds can't be carried out for all the received messages, leaving the only viable option of doing verify-on-demand, i.e. verify only those messages that would result in an alert. This puts a premium on fast decoding, as all the incoming messages need to be decoded first irrespective of whether they are verified or not. Decode times are less important if the signature verification is done in hardware, which can be several orders of magnitude faster.

Existing Approach of 1609.2

Any specification that describes how data is to be structured and encoded needs a definition of a syntax that may be used to specify the data structures. This syntax is referred to as, among others, a presentation language [1], a specification language [17], and a syntax notation (implicit in the name of ASN.1 itself). In the rest of this paper we use the terminology "presentation language".

The presentation language used in 1609.2-2013 is based on that defined in IETF RFC 5246 [15], where it is described as a "very basic and somewhat casually defined presentation syntax". Informally, it has the following syntax elements:

- `uint8, uint16, uint24, uint32` – unsigned integer types
- `enum` – enumerated types
- `opaque` – octet strings with length specified as described below
- Fixed length vectors: `TOld TNew[n]` is a vector named `TNew`, containing one or more elements of type `TOld`, with total length n octets (note, not the same as containing n instances of a `TOld`).
- Variable-length vectors: `TOld TNew<2^8-1>` is a vector named `TNew`, containing one or more elements of type `TOld`, with total length no more than `2^8-1` octets, i.e. such that the total length can be encoded in one octet. Vectors can be defined with any other maximum length, but the most natural lengths are `<2^8k-1>` for some positive integer `k`, as these map to length encodings in integer numbers of octets. For more flexibility, the spec also permits:
- Variable-length vectors with variable length lengths: `TOld TNew<var>` is a vector named `TNew`, containing one or more elements of type `TOld`, with total length encoded as:
    - One octet with top `bit = 0` if less than `2^7-1`;
    - Two octets with top bits of the first `octet = 10` if less than `2^14-1`;
    - Three octets with top bits of the first `octet = 110` if less than `2^21-1`;
    - … and so on

  Note that "variable-length vectors with variable length lengths" doesn't exist in pure TLS but is an addition to the TLS notation.

Structures with multiple elements may be defined using C-like syntax, for example:

```
struct {
      Type1 name1;
      Type2 name2;
  } StructType;
```

Optional elements may be handled in two different ways: selector values with a select statement, or flags values with a statement from the `if_set` family. This allows structures to be specified in the way that will lead to the most compact encoding: if the selector values are naturally integers from a range or enumerated values, `select` leads to a more compact encoding; if the selector values are more naturally considered a bitmap, the use of `flags` leads to a more compact encoding.

Extensibility is provided by use of an `other_value` selector value in a select statement, and by an `if_other_value_set` selection operator when using flags.

Data structures produced with 1609.2 are always octet-aligned, allowing for fast encoding and decoding.

ASN.1 Background

Abstract Syntax Notation 1 (ASN.1) is a standard [11] that defines a presentation language for specifying and implementing protocols, specially communications and computer networking. One of the most attractive and useful features of ASN.1 is the ability to specify the protocol independent of the underlying implementation and representation choices.

ASN.1 was introduced in 1984 as part of CCITT X.409:1984 [29], but has undergone several major updates since. In 1988, support for X.509 digital certificates [9] was added, and 7 years later in 1995

additional improvements were made to support bandwidth and CPU-constrained devices as part of X.680 [11]. Most recently in 2002, support for XML was added to ASN.1 as part of X.690 [12]. Due to its versatile nature, today ASN.1 is almost ubiquitous [30], is deployed in well over a billion devices, including telephone systems, world wide web, electronic mail, and aviation control systems.

Encoding rules. In ASN.1, the definition of the logical structure is separated from the definition of the encoding rules. The logical structure determines which data elements are included and may be used to indicate the order in which they appear. The encoding rules determine how the data elements are converted into bit or octet strings for transmission.

A number of different encoding rules are specified in the ASN.1 base standards.

- The Basic Encoding Rules (BER) [12] produce byte-aligned output but with significant overhead due to nested length fields and including a type identifier on each field. BER also allows multiple valid encodings for any given data structure. This creates issues for cryptographic processing as cryptographic operations such as signing are carried out on byte arrays, not on the underlying data structure, so an unambiguous encoding method is necessary to ensure that the same data always results in the same byte array and hence in the same hash value and consistent signatures.
- The Distinguished Encoding Rules (DER) [12] are a profile of the Basic Encoding Rules that give a single valid encoding for each data structure. They have issues similar to BER in producing significant encoding overhead.
- The Packed Encoding Rules (PER) [13] were defined for use in cases where transmitted packet size is a critical concern. In PER, encoded data elements are treated as bit arrays rather than octet arrays and are concatenated bitwise. This allows for compact encodings, but at the cost of requiring significantly more bit manipulations when encoding and decoding. There are two variants of PER, PER Unaligned and PER Aligned. PER Unaligned is more aggressive than PER Aligned at packing data as compactly as possible (with the exception of small integer values, which for some reason are packed more compactly in PER Aligned).
- The Octet Encoding Rules (OER) [14] are a more recent set of encoding rules, intended to produce octet-aligned output like DER while at the same time avoiding some of the overhead that comes with the use of DER. OER allows for rapid encoding and decoding.

1609.2 v ASN.1: Disadvantages of 1609.2

Disadvantages of the approach to optional elements include:

1) **Syntax mixed with encoding**. The syntax (the logical structure of the data types) is inseparable from the encoding rules, in contrast to ASN.1.
2) **Less compact specifications if multiple selector values**. If there are multiple possible values for a selector, the 1609.2 presentation language needs at least two lines to specify the behavior for each value, as opposed to the ASN.1 CHOICE type that needs only one.
3) **Less compact specifications for optional fields**. If a structure contains multiple optional fields, the 1609.2 presentation language requires that the structure has a named selector for each field as well as the field itself, or that it uses flags. For example, the following ASN.1 and 1609.2 structures denote the same information. The 1609.2 structure is more complicated to express, harder to read, and takes more space.

```
1609.2: flags {has_cat(0), has_horse(1), has_badger(2)}
HasPetFlags;
        struct {
            HasPetFlags pet_flags;
            if_set      (pet_flags,      has_cat)      {
                opaque cat_name<var>;
            }
            if_set     (pet_flags,      has_horse)     {
                opaque horse_name<var>;
            }
            if_set     (pet_flags,     has_badger)     {
                opaque badger_name<var>;
            }
        } PetNames;

ASN.1:  PetNames ::= SEQUENCE {
                OCTET STRING CatName OPTIONAL;
                OCTET STRING HorseName OPTIONAL;
                OCTET STRING BadgerName OPTIONAL;
        }
```

Political disadvantages: Work on 1609.2 is partially sponsored by the US Department of Transportation, which has a preference that standards that it sponsors are expressed in ASN.1 (or other standard presentation languages) where possible.

Availability of tools: There are many commercial [18] [19] [20] and open-source [21] [22] compilers for ASN.1, i.e. tools that take an ASN.1 schema and an indication of a set of encoding rules and produce source code for encoders and decoders for those schemas. There are no such tools for 1609.2 / TLS.

Completeness of presentation language: It can be shown that in ASN.1, any syntactically correct definition will be unambiguous, and any ASN.1 schema can be checked for syntactic correctness [23]. No such proof exists for the 1609.2 presentation language.

Related Projects

Comparisons of ASN.1 with other encoding schemes have been carried out for many years: for example, [17] is a comparison of ASN.1 and XML dating from 2004. In the cooperative ITS world, there are a number of related projects. ETSI TS 103 097 [24] is currently using the TLS-derived presentation language that is used in the current version of 1609.2, and has no plans to change. Within ETSI, the CAM [25] and DENM [26] messages are currently defined using ASN.1 and encoded with PER Unaligned; the corresponding US standard, SAE J2735 [5], which defines the Basic Safety Message (BSM), is defined in ASN.1 but uses a “blob” format for some key structures. A project parallel to this one is underway to determine whether the blob format should be replaced with PER Unaligned. Finally, the Security Credential Management System (SCMS) [27] being defined for use with the US mandated system is defining interfaces using ASN.1.

The mandate in the US is expected to require that devices send BSMs as defined in [5], signed and encapsulated in a 1609.2 SignedData structure. The use case of signed BSMs is therefore the primary focus of our investigation, as these will constitute the vast majority of time-critical messages received.

Description of Project

Definition of ASN.1 Schema

We started by defining a candidate ASN.1 schema. This was primarily based on 1609.2, but also incorporated concepts and fields from ETSI TS 103 097 [24] (hereafter "103 097"), which contains the data types used for secure communications in the European Cooperative ITS system. Ideas from 103 097 were incorporated for two reasons: (a) to propose a schema that might be acceptable for use in both Europe and the US in some future harmonization process, and (b) because those ideas supported use cases that the 1609 working group considered useful in the US even in the absence of harmonization.

In designing the schema we took into account some key approaches of 1609.2. **Single-pass processing** is the property that the encoded message can be produced and processed in a single pass by both sender and receiver. For signed messages, this means the message must start with an indicator of at least the hash function as the receiver needs to know this to generate the hash for verification, followed by the message contents, followed by the signature as this is generated last by the sender. For encrypted messages, this means the public-key encrypted bulk encryption key must appear first, followed by the encrypted payload. 1609.2 also enforces **unambiguous ordering of fields** and **no duplicates**: these principles are easy to obtain in ASN.1, but in 103 097 are enforced not by the data structures themselves but by the standards text surrounding them. We considered it important to have the machine-parseable information, i.e. the data structures, contain as much information as possible about what is a valid structure.

The resulting schema is in the full version report that is available upon request from the authors.

Selection of Encoding Rules and Definition of Alternative ASN.1 Schema

For performance measurement it is necessary to select a set of encoding rules to be used. We were interested in the performance of two different sets of encoding rules: PER Unaligned and OER. PER Unaligned produces in general the smallest encodings, and is specified for use in other Intelligent Transportations System (ITS) standards such as CAM [25] and DENM [26] within ETSI. OER allows for faster encoding and decoding rules.

Alternative ASN.1 schema to maximize octet alignment with the PER. The original ASN.1 schema is designed with high-level design principles in mind, as noted above: one-pass processing, unambiguous ordering of fields, no duplicates. This does not naturally produce octet-aligned fields when encoding or decoding with PER Unaligned. However, octet aligned fields are desirable for two reasons. First, the 1609.2 processing involves some operations that are naturally octet-aligned: specifically, signing and encryption. Both the signing and the encryption operations are carried out on data that is then encapsulated in a larger data structure. If these data are not encoded such that they start at an octet boundary, they will need to be realigned to start on an octet boundary in order to be used with most cryptographic libraries, which are octet-aligned. The cost of bit-shifting and memory allocation to support this is a potential concern. Second, even without the octet-alignment necessary to support cryptographic operations, octet alignment makes encoding and decoding operations faster. We therefore defined an alternative schema under which, with PER Unaligned, more of the encoded output will be octet aligned. We call this the "P" schema (for "oPtimized for PER Performance").

The P schema does not meet the design principles outlined above. For example, compare the definitions of SignedData in the two schemas:

```
Original:
SignedData ::=
  SEQUENCE{ signer              SignerIdentifier,
            unsignedData        ToBeSignedData,
            signature           Signature
          }

P Schema:
SignedData ::=
  SEQUENCE{ spacer              Uint3,
            signature           Signature,
            unsignedData        ToBeSignedData,
            signer              SignerIdentifier
          }
```

The P-Schema uses the fact that an encoded signature will be an integer number of bytes preceded by 5 bits of header information, so to preserve octet alignment the signature is moved to the start, preceded by a 3-bit "spacer" field. This ensures that the payload of the signature is octet aligned, and also that the ToBeSignedData is octet aligned. However, the P-schema is less desirable than the original in three ways: first, it produces larger encoded structures (because in order to obtain octet alignment fields must be shifted to the right, making the overall length larger); second, it does not preserve one-pass processing (in this example, because the signature comes at the start); third, it requires the use of explicit spacer fields. The spacer fields are undesirable because (a) they mean the ASN.1 structures contain encoding information as well as logical information, contrary to the intent of ASN.1, and (b) for any future version that changes the contents of these structures, the designer will have to manually check, and if necessary, adjust the size of the spacer fields.

The P-Schema is therefore second best to the original schema except in so far as it may allow for improved performance. In our experiments we tested performance of both the original and the P-Schema to see whether the performance gains from use of the P-Schema were significant enough to outweigh the other disadvantages.

Selection of Candidate Messages for Encoding/Decoding

We created 64 different filled data structures, SecuredData-001 to SecuredData-064, based on the original ASN.1 schema. Since there are a large number of optional fields in 1609.2, the idea was not to be exhaustive but to provide a wide range of encoding complexities. The data structures were created by incrementally adding optional fields to an empty data structure, producing successively larger data structures. The data structures tested do not necessarily represent structures that are under consideration for use in real settings, but

they cover the range of possible sizes and complexities of data structures. The data structures are in the full version report that is available upon request from the authors.

We then created the 64 equivalent messages based on the P-Schema. This was straightforward because the only differences between the P-Schema and the original schema are ordering and the inclusion of spacer fields. These messages are not explicitly presented in this paper but are easy to recreate.

Finally, we created the 64 equivalent data structures based on 1609.2. The definition of equivalent messages was not straightforward sometimes, as the ASN.1 schema used fields that were not included in the 1609.2 structures or vice versa. In some cases, this was a result of 1609.2 explicitly including selector fields in the data structures while with ASN.1 the selector fields are provided automatically by the encoding rules and not explicitly stated. In other cases, the difference was due to the fact that the ASN.1 schema incorporated ideas from 103 097 rather than attempting to be a direct translation of 1609.2. The 1609.2 data structures are in the full version report that is available upon request from the authors.

Performance Measurements

The messages were tested for encoded message sizes, and encoding and decoding times. For each data structure, the encode and decode operations were run enough times that the total running time was several seconds, and then the time per operation was calculated from that.

In selecting the platforms for comparison, several factors (e.g., cost and ease of availability) came into play. The primary platforms for the comparison were:

- Two Linux PCs, one running at 1.9 GHz, the other running at 3.2 GHz
- Cohda MK4, ARM A9 platform at 800 MHz (similar to [28])

The intent in using the Cohda box was not to favor one particular supplier, but to use a box that seems to be a good example of the OBEs used in Safety Pilot Model Deployment [31]. We approached other OBE suppliers but, for reasons of time, when Cohda agreed to provide an OBE we did not pursue obtaining similar devices from other vendors. We consider the results on the Cohda box to be broadly representative of what results can be expected to be on other modern prototype equipment.

We engaged two ASN.1 suppliers:

- OSS Nokalva [18], who ran tests with both the original schema and the P-Schema, using PER Unaligned for both schemas and OER for the original schema. OSS Nokalva provided two sets of performance figures for each (schema/encoding rules) combinations. The first set of performance figures was for code produced by their off-the-shelf compiler. The second was for code that was hand-optimized, taking the off-the-shelf output as a starting point. These figures are indicated by "OTS" and "Custom", respectively, in the results tables below.
- Marben Products [19], who ran tests with the P-Schema and PER Unaligned only. The Marben figures were obtained using code produced by their off-the-shelf compiler only.

Security Innovation ran the performance tests for the 1609.2 data structures. The 1609.2 tests were run on the same machines as the OSS tests were run on, to improve the comparability of the results.

For 1609.2, Security Innovation tested two implementations: a full-featured C++ implementation that converts between encoded octet strings and instantiated objects, and a stripped-down decode-only C implementation that converts from octet strings into C structures. The C++ implementation proved to have significant overhead due to object creation and memory management. The C implementation was significantly faster, and also was a more suitable basis for comparison to the ASN.1 encoder/decoders provided by the ASN.1 suppliers, as those decoders also output C structures. We therefore used the C decoder as the basis for performance comparison. The C implementation did not create encodings, so we omitted encoding times from the performance comparison. We consider this omission reasonable as decoding is a much more common event than encoding, so encoding/decoding times on any given device will be dominated by the decoding times.

Results Summary

Here we present a summary of our results; detailed results are in the full version report that is available upon request from the authors. The summary gives results for eight of the 64 data structures, selected to cover a range of message types and sizes. Table 1 shows the data structures picked for presentation and the friendly names given to them.

None of the data structures that were tested by the ASN.1 vendors corresponded exactly to the Basic Safety Message [5], which is the main structure of interest, so we added extrapolated results for the BSM as follows:

- We defined two new messages: BSM-digest and BSM-cert, that closely resemble a Basic Safety Message (BSM) with a digest and a certificate, respectively.
- We measured the encoded size exactly with ASN.1 tools and with Security Innovation's 1609.2 implementation.
    - Since the experiments were carried out, some proposals have been made for changes to the schema; we refer to the resulting schema as the "2014 schema". We present below figures for the encoded size for BSMs under the 2014 schema as well as under the original schema and P-schema.
- We measured the decode times exactly for 1609.2.
- We performed extrapolations of the decode times for ASN.1 based on the measured results using two options: (a) linear fit w.r.t. message size and (b) linear fit w.r.t. number of optional fields included. At a high level, the extrapolations were carried out as follows. For a given (vendor, encoding rules, platform) combination:
    - Do the linear regression for size on the 64 messages
    - Do the linear regression for number of fields on the 64 messages
    - Between the above two options, select the one with better R^2
    - Use that linear rule for the extrapolation

For most of the (vendor, encoding rules, platform) combinations, the number of fields fit had a significantly better R^2 value than that of the size fit.

For the resulting 10 messages, we present encoded message sizes in Table 2, decoding times on Linux desktop with results normalized to a clock speed of 3 GHz in Table 3, and decoding times on Cohda box in Table 4. The encoded message sizes are in bytes, and the decoding times are in nanoseconds.

Finally, as noted in the Background section, decoding performance is very important for V2X communication, and therefore we also present decoding time projections for some realistic scenarios in Table 5. We pick two scenarios: (a) 100% of the messages have full certificates, (b) only 20% of the messages have full certificates and the rest 80% have certificate digests that are much small. For both these scenarios, we test decoding times for 100 messages per second and 1000 messages per second, resulting in a total of 4 different scenarios. The decoding times are reported in microseconds.

Table 1: Mapping of Message Names

#	Data structure	Friendly name
1	SecuredData-001	Unsecured
2	SecuredData-003	Signed-digest-small
3	SecuredData-006	Signed-digest-medium
4	SecuredData-023	Signed-digest-large
5	SecuredData-025	Signed-cert-small
6	SecuredData-033	Signed-cert-medium
7	SecuredData-058	Signed-cert-large
8	SecuredData-062	Encrypted

Table 2: Encoded Message Sizes (Bytes)

	ASN.1 – Vendor 1	**ASN.1 – Vendor 2**			**1609.2**
	UPER-P	UPER-O	UPER-P	OER	
Unsecured	19	19	19	19	19
Signed-digest-small	116	114	116	117	80
Signed-digest-medium	285	283	285	285	281
Signed-digest-large	322	320	322	323	299
Signed-cert-small	428	426	428	438	404
Signed-cert-medium	445	443	445	456	419
Signed-cert-large	780	778	780	813	564
Encrypted	100	99	100	103	102
BSM-digest#	182	180/177*	182	183/182*	176
BSM-cert#	255	253/240*	255	265/256*	247
-O: original schema, -P: P-schema, *: 2014 schema					

Table 3: Decoding Times on Linux Desktop Normalized to 3GHz (nsec)

	ASN.1 – Vendor 1	**ASN.1 – Vendor 2**			**1609.2**
	UPER-P OTS	UPER-O OTS/Custom	UPER-P OTS/Custom	OER OTS/Custom	
Unsecured	770	109/44	92/38	71/15	23
Signed-digest-small	1320	364/165	272/129	165/47	201
Signed-digest-medium	1650	531/243	333/149	198/51	230
Signed-digest-large	2090	531/257	474/232	257/75	137
Signed-cert-small	2970	926/470	829/409	417/134	308
Signed-cert-medium	3300	981/502	874/424	455/144	307
Signed-cert-large	6380	2351/1227	2098/1093	1074/386	560
Encrypted	1430	394/189	286/136	225/62	90
BSM-digest#	1657	406/189	328/153	202/53	153
BSM-cert#	2717	821/413	708/355	385/121	223
-O: original schema, -P: P-schema, #: extrapolated values					

Table 4: Decoding Times on Cohda Box (nsec)

	ASN.1 – Vendor 1	**ASN.1 – Vendor 2**			**1609.2**
	UPER-P OTS	UPER-O OTS/Custom	UPER-P OTS/Custom	OER OTS/Custom	
Unsecured	1700	771/316	692/252	655/97	306
Signed-digest-small	2900	2427/1315	2141/935	1315/316	1450
Signed-digest-medium	3400	3457/2113	2677/1052	1599/371	1572
Signed-digest-large	4200	3680/1884	3569/1770	1998/524	1905
Signed-cert-small	6500	6971/3680	6535/3123	3301/1023	4207
Signed-cert-medium	7000	7624/3904	6971/3212	3457/1081	4393
Signed-cert-large	14700	17550/8986	16486/8169	8441/3034	7928
Encrypted	3000	2677/1370	2213/935	1644/335	1316
BSM-digest#	3402	2918/1464	2576/1130	1564/358	1471
BSM-cert#	5808	6002/3191	5547/2637	2970/910	2270
-O: original schema, -P: P-schema, #: extrapolated values					

Table 5: Decoding Time Projections for Realistic Scenarios on Cohda Box (µsec)

	ASN.1 – Vendor 1	**ASN.1 – Vendor 2**			**1609.2**
	UPER-P OTS	UPER-O OTS/Custom	UPER-P OTS/Custom	OER OTS/Custom	
100 msg / sec (20% cert, 80% digest)	388	353/181	317/143	185/47	163
100 msg / sec (100% cert)	581	600/319	555/264	297/91	227
1000 msg / sec (20% cert, 80% digest)	3883	3534/1809	3170/1431	1845/469	1631
1000 msg / sec (100% cert)	5808	6002/3191	5547/2637	2970/910	2270
-O: original schema, -P: P-schema					

Conclusion

The size differences between the PER Unaligned and the OER encodings are not significant; both are slightly larger (see Table 2) than the corresponding 1609.2 encodings in almost all cases. In the

most taxing realistic scenario (1000 messages per second with 100% full certificate, i.e. last line of Table 5), we show that 1609.2 decoding takes about 0.2% of the available time, while OER and UPER-O take about 0.1-0.3% and 0.3-0.6% of the available time, respectively. This shows that even though the relative difference between 1609.2 and UPER-O is significant, no encoding method is likely to take a significant amount of real-world time in practice, as they are all less than 1% of the available time.

The other advantages of ASN.1 – availability of tools, conformance with policy, ease of testing – are sufficiently significant that the 1609 working group had taken an indicative vote that 1609.2 should move to ASN.1 unless ASN.1 has significant performance cost. The results of this performance analysis show that the performance impact of ASN.1 is minimal. The 1609.2 working group therefore voted in the October 2014 meeting to use ASN.1 with the OER as the presentation language and encoding rules for future versions of 1609.2.

References

[1] IEEE Standard Wireless Access in Vehicular Environments — Security Services for Applications and Management Messages, IEEE Standard 1609.2-2013, 2013. Available: http://standards.ieee.org/findstds/standard/1609.2-2013.html

[2] IEEE Standard for Wireless Access in Vehicular Environments (WAVE) — Networking Services, IEEE Standard 1609.3-2010, 2010. Available: http://standards.ieee.org/findstds/standard/1609.3-2010.html

[3] IEEE Standard for Wireless Access in Vehicular Environments (WAVE) — Multi-channel Operation, IEEE Standard 1609.4-2010, 2010. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=5712767

[4] IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11-2012, 2012. Available: http://standards.ieee.org/about/get/802/802.11.html

[5] Dedicated Short Range Communications (DSRC) Message Set Dictionary, SAE Standard J2735_200911, 2009. Available: http://standards.sae.org/j2735_200911/

[6] Dedicated Short Range Communication (DSRC) Minimum Performance Requirements, SAE Standard J2945, 2010. Available: http://standards.sae.org/wip/j2945/

[7] U.S. Department of Transportation - National Highway Traffic Safety Administration. Decision to Move Forward with Vehicle-to-Vehicle Communication Technology for Light Vehicles. Available: http://www.nhtsa.gov/About+NHTSA/Press+Releases/2014/USDOT+to+Move+Forward+with+Vehicle-to-Vehicle+Communication+Technology+for+Light+Vehicles

[8] U.S. Department of Transportation - National Highway Traffic Safety Administration. Advance Notice of Proposed Rulemaking to Begin Implementation of Vehicle-to-Vehicle Communications Technology. Available: http://www.nhtsa.gov/About+NHTSA/Press+Releases/NHTSA-issues-advanced-notice-of-proposed-rulemaking-on-V2V-communications

[9] Public-Key Infrastructure (X.509), IETF PKIX Working Group, 2013. Available: http://datatracker.ietf.org/wg/pkix/documents/

[10] R. Housley, "Internet Engineering Task Force (IETF) Request for Comments (RFC) 3852: Cryptographic Message Syntax (CMS), 2004". Available: http://www.ietf.org/rfc/rfc3852.txt

[11] Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation, ITU-T Recommendation X.680 (11/2008), 2008. Available: http://handle.itu.int/11.1002/1000/9604

[12] Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), ITU-T Recommendation X.690 (07/2002), 2002. Available: http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf

[13] Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER), ITU-T Recommendation X.691 (07/2002), 2002. Available: http://www.itu.int/ITU-T/studygroups/com17/languages/X.691-0207.pdf

[14] National Transportation Communications for ITS Protocol — Octet Encoding Rules (OER) Base Protocol. Joint Standard of AASHTO, ITE, and NEMA, version 01.15. NTCIP 1102:2004, 2005. Available: https://www.nema.org/Standards/ComplimentaryDocuments/NTCIP1102.pdf

[15] T. Dierks, E. Rescorla, "Internet Engineering Task Force (IETF) Request for Comments (RFC) 5246: The Transport Layer Security (TLS) Protocol Version 1.2". Available: http://tools.ietf.org/html/rfc5246

[16] Code Climate, 5 Reasons to Use Protocol Buffers Instead of JSON For Your Next Service, http://blog.codeclimate.com/blog/2014/06/05/choose-protocol-buffers/

[17] D. Mundy and D. Chadwick, "An XML Alternative for Performance and Security: ASN.1". In IEEE Computer Society, IT Professional, Jan/Feb 2004. Available: http://kar.kent.ac.uk/14031/1/AnxmlMun.pdf

[18] OSS Nokalva, ASN.1 Tools. Available: http://www.oss.com/asn1/products/asn1-products.html

[19] Marben, ASN.1 Tools. Available: http://www.marben-products.com/asn.1/asn1-tools-overview.html

[20] Objective Systems, ASN.1 Tools. Available: http://www.obj-sys.com/products/asn1c/

[21] Institute For Information Industry, ASN.1 Tool. Available: http://iiiasn1.sourceforge.net

[22] L. Walkin, ASN.1 Open Source Software. Available: http://lionet.info/asn1c/compiler.html

[23] A. Triglia, "ASN.1 for More Effective Network Standards". In IEEE 802 Plenary Tutorials, 2010. Available: http://www.ieee802.org/802_tutorials/2010-11/Alessandro%20Triglia%20-%20ASN.1%20Tutorial%20Dallas%20-%2020101108T1701.pptx

[24] Security header and certificate formats, ETSI Technical Specification 103 097, Version 1.1.1, 2013. Available: http://www.etsi.org/deliver/etsi_ts/103000_103099/103097/01.01.01_60/ts_103097v010101p.pdf

[25] Intelligent Transport Systems (ITS), Vehicular Communications, Basic Set of Applications, Part 2: Specification of Cooperative Awareness Basic Service, Draft ETSI EN 302 637-2 V1.3.0, 2013. Available: http://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.00_20/en_30263702v010300a.pdf

[26] Intelligent Transport Systems (ITS), Vehicular Communications, Basic Set of Applications, Part 3: Specifications of Decentralized Environmental Notification Basic Service, Draft ETSI EN 302 637-3 V1.2.0, 2013. Available: http://www.etsi.org/deliver/etsi_en/302600_302699/30263703/01.02.00_20/en_30263703v010200a.pdf

[27] W. Whyte, A. Weimerskirch, V. Kumar and T. Hehn, “A Security Credential Management System for V2V Communications”. In IEEE Vehicular Networking Conference (VNC), 2013, pp. 1 – 8. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6737583
[28] Cohda Wireless, MK4a evaluation kit. Available: http://cohdawireless.com/Portals/0/PDFs/CohdaWirelessMK4a.pdf
[29] CCITT Draft Recommendation X.409: Message Handling Systems. Presentation Transfer Syntax and Notation CCITT, 1984.
[30] Application fields of ASN.1. Available: http://www.itu.int/en/ITU-T/asn1/Pages/Application-fields-of-ASN-1.aspx
[31] Safety Pilot: Model Deployment. Available: http://safetypilot.umtri.umich.edu
[32] G. Bansal and J. Kenney, “Controlling Congestion in Safety-Message Transmissions: A Philosophy for Vehicular DSRC Systems”. In IEEE Vehicular Technology Magazine, December 2013.

Contact Information

Virendra Kumar: vkumar@securityinnovation.com

William Whyte: wwhyte@securityinnovation.com

Acknowledgments

Funded under HWSA Task Order “Continued Support to Dedicated Short Range Communications (DSRC) Standards Development and Related Efforts”, Task Order: DTFH61-10-D-00015-T-130006.

Thanks to OSS Nokalva, Marben, Crash Avoidance Metrics Partnership, and the IEEE 1609 working group.