

Performance Analysis of Existing 1609.2 Encodings v ASN.1

Virendra Kumar and William Whyte
Security Innovation

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.

CITATION: Kumar, V. and Whyte, W., "Performance Analysis of Existing 1609.2 Encodings v ASN.1," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 8(2):2015, doi:10.4271/2015-01-0288.

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

1. V2V and V2I are together referred as V2X.

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 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 1/10th 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<28-1>` is a vector named `TNew`, containing one or more elements of type `TOld`, with total length no more than `28-1` 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 `<28k-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 `27-1`;
 - Two octets with top bits of the first octet = 10 if less than `214-1`;
 - Three octets with top bits of the first octet = 110 if less than `221-1`;
 - ... 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.