Linux-Stack Based V2X: White Paper

Duncan Woodbury, Nicholas Haltmeyer, and Robert Leale
{p3n3troot0r, ginsback}@protonmail.com, robertleale@canbushack.com

May 13, 2017

Abstract

Vehicle-to-vehicle (V2V) and, more generally, vehicle-to-everything (V2X) wireless communication enables semi-autonomous driving via the exchange of state information between a network of connected vehicles and infrastructure units. Following 10+ years of standards development, particularly the IEEE 802.11p and 1609 family, a lack of available implementations has prevented the involvement of the security community in development and testing of these standards. Analysis of the WAVE/DSRC protocols in their existing form reveals the presence of vulnerabilities which render the protocol unfit for use in safety-critical systems. We present a complete Linux-stack based implementation of IEEE 802.11p and 1609.3/4 which provide a means for hackers and academics to participate in the engineering of secure standards for intelligent transportation.

1 Introduction

As vehicular ad-hoc networks (VANETs) will serve the purpose of disseminating messages related to critical safety procedures, effective security is vital. A network compromise would allow an adversary to launch a wide array of attacks. Current standards defining the implementation and use of VANETs (IEEE 1609 – WAVE) use a certificate management system to assure the confidentiality and authenticity of messages, as well as a misbehavior reporting system to maintain a network-wide distributed trust of participants. If an attacker is able to compromise a peer, they have the ability to launch a variety of attacks. Examples of these threats are as follows:

- Distributing false safety messages - Allows the attacker to obstruct or divert traffic by leading vehicles to believe there is an upcoming accident or road block, beacon as an emergency vehicle, or maliciously alert the driver of a fictitious imminent crash. The effects of this can lead to significant cost in traffic congestion or even loss of life.
- Masquerading as a toll station - Allows the attacker to automatically collect payment information from passing vehicles.
- Issuing false certificates - Allows the attacker to grant privileges and proliferate any of the above attacks to colluding vehicles.
- Issuing false certificate revocations - Allows the attacker to effectively deny service until new certificates can be authorized.
- Distributing false misbehavior reports - Allows the attacker to either grant undue trust to malicious vehicles or deny service (and lead to the certificate revocation of) well-behaved vehicles.

1.1 Contributions

We present a Linux kernel implementation of the IEEE 802.11p and 1609.3/4 standards, encompassing all features related to transmission/reception, channel switching, and message encoding/decoding, including a userspace utility facilitating the use of V2X with the SAE J 2735 basic message dictionary to communicate with other Linux boxes and proprietary DSRC radios.

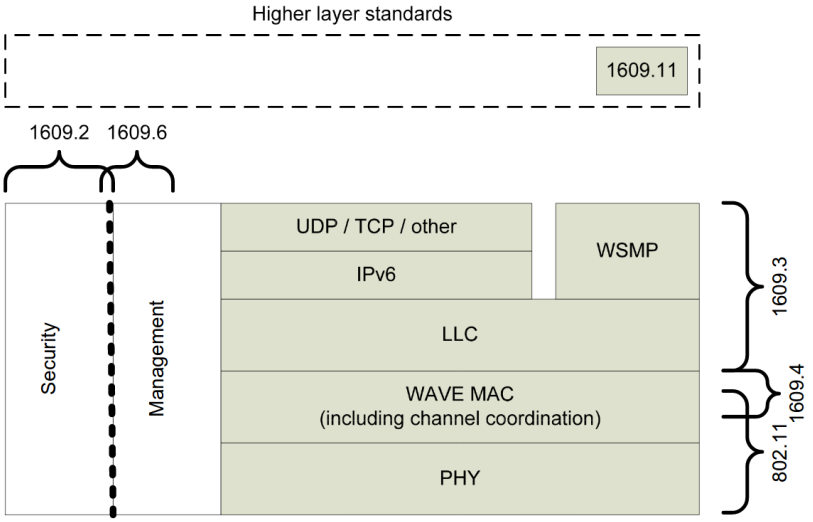

Figure 1: WAVE protocol stack [2].

1.2 Wireless Access in Vehicular Environments

Wireless Access in Vehicular Environments (WAVE) is a set of standards that form a comprehensive framework for vehicle-to-vehicle communication. The base of the protocol is defined in IEEE 802.11p and defines operation on the 5.85 to 5.925GHz band [1].

The carrier wave structure is highly similar to that defined by 802.11p, using orthogonal frequency division multiplexing (OFDM). The remainder of the specification is defined in IEEE 1609 and deals with message structure, security services, message aggregation, and forwarding within the ad-hoc network. Figure 1 shows the OSI-like structure of WAVE.

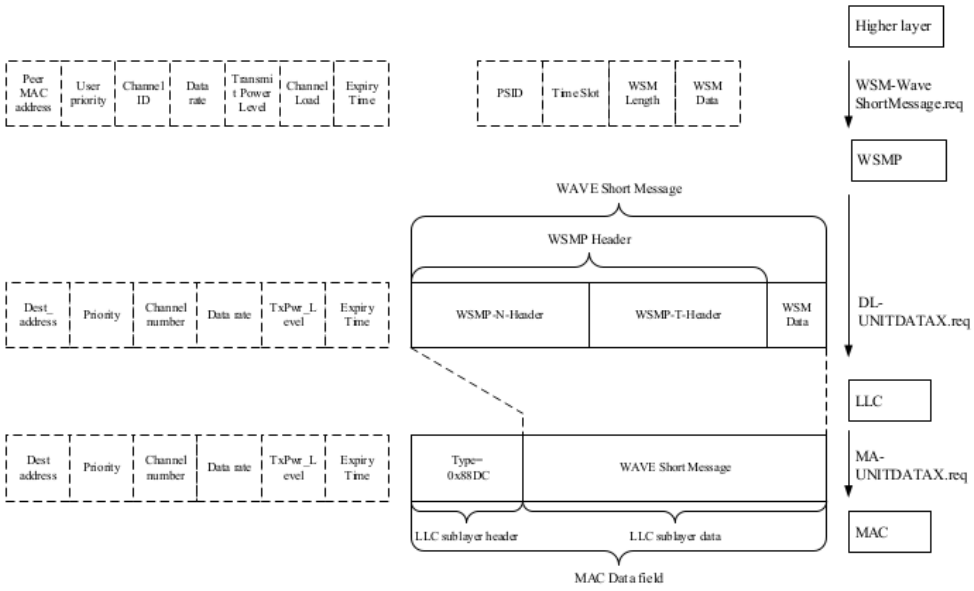

Figure 2: WAVE Short Message specification [4].

Message encoding is defined in the WAVE Short Message (WSM). The collection of data structures and management entities involved in manipulating WSMs is the WAVE Short Message Protocol [2, 3, 4, 5]. See Figure 2 for a breakdown of WSM data fields. Additionally, WAVE Service Advertisements are broadcast on a fixed interval, notifying peers of the car's availability. See Figure 3 for a breakdown of WSA data fields, and Figure 4 for WSA encapsulation within the

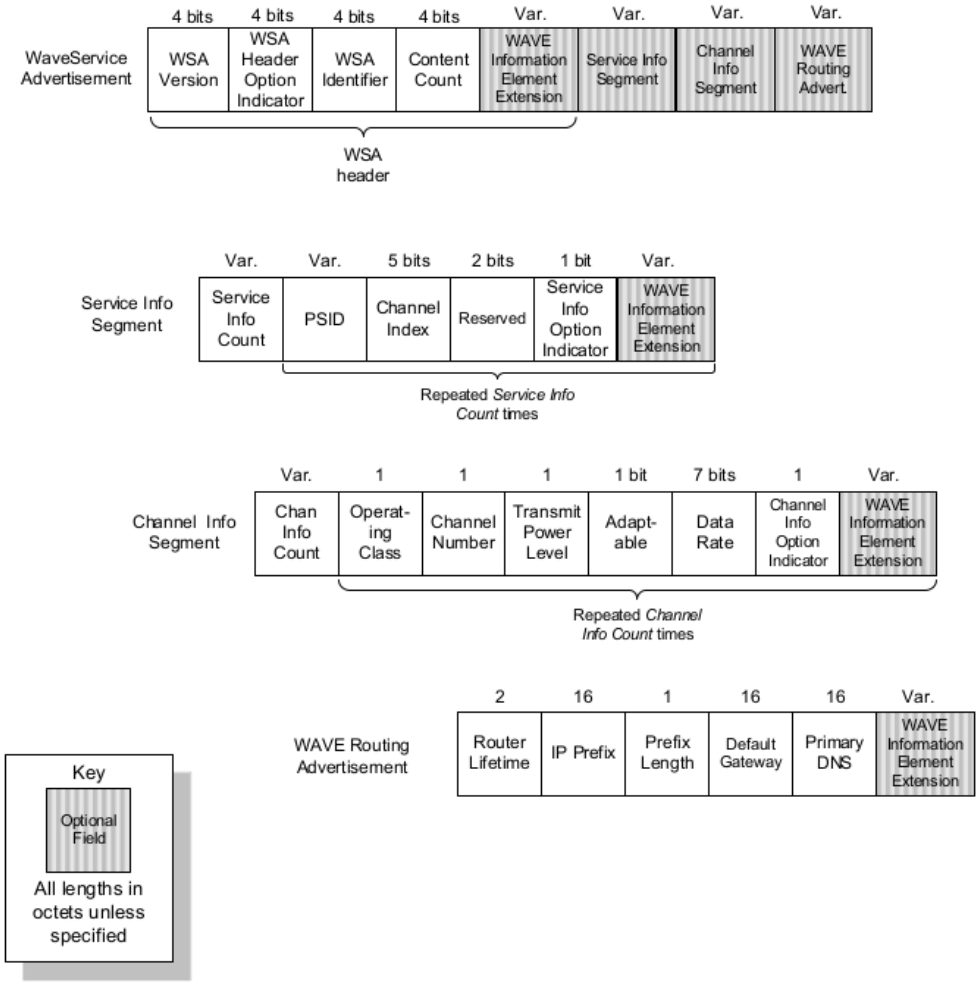

Figure 3: WAVE Service Advertisement specification [4].

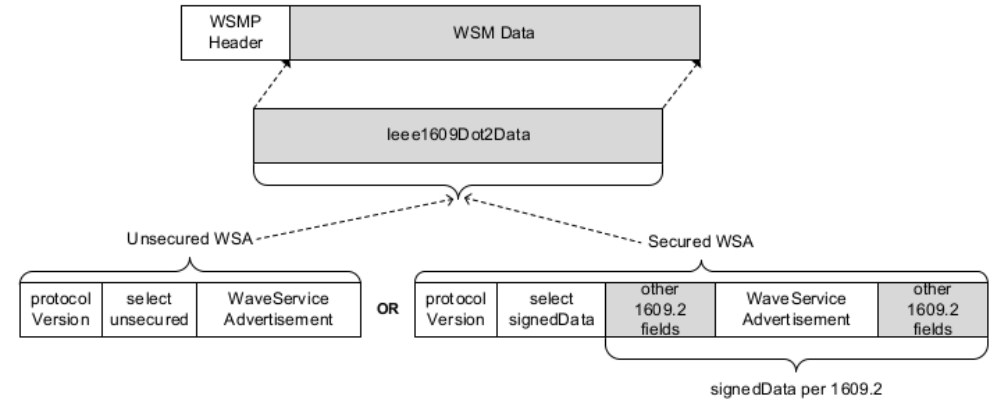

Figure 4: WSA encapsulation in a WSM [4].

WSM.

1.3 Channel Switching

WAVE subdivides the bandwidth into 7 channels of 10MHz each. Channels are devoted to non-safety messages, traffic efficiency messages, and critical safety messages. Channels marked for longer range communications have a higher transmission power. The full range of channel settings can be seen in Figure 5. As WAVE-capable devices need only use a single antenna, a switching method is defined to receive control channel (CCH) and service channel (SCH) data frames at intervals of 50ms each. A diagram of this switching procedure can be seen in Figure 6.

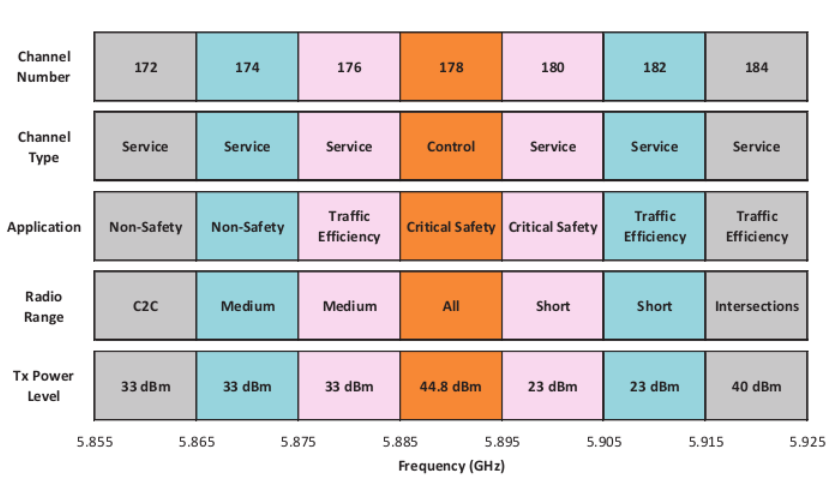

Figure 5: Channels subdivision and parameters [10].

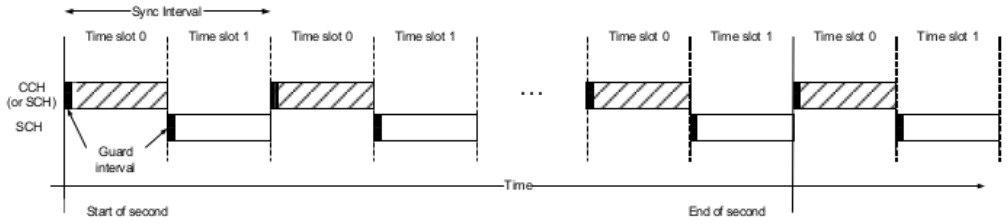

Figure 6: Channel switching procedure [5].

1.4 WAVE Security

Security in WAVE is defined by IEEE 1609.2. This enables encryption and signing through a certificate system. Certificates are chained against a trust anchor that demonstrates the authenticity of each peer in the network. The central certificate authority acts as this trust anchor. The final certificate is predicated on the validity of previous certificates. Each certificate has a set of associated permissions.

WAVE supports a peer-to-peer certificate discovery mechanism. Using this, vehicles and roadside units (RSUs) can request nearby peers to provide certificate information in the event that the full chain cannot be verified, due to a lack of information. Upon being given all relevant data, the management service can then determine if the certificate chain in question is valid.

2 Implementation

Our implementation is designed to integrate seamlessly with the Linux kernel networking subsystem. The implementation of IEEE 802.11p works through modification of the `mac80211`, `cfg80211`,

and `nl80211` standard Linux networking utilities, and by making very limited modifications to a wireless hardware driver. In this case support is given for the Atheros `ath9k` and Realtek `rtlwifi` drivers and compatible devices. Note that some effort was made to add support for IEEE 802.11p to the mainline Linux kernel in 2014, although full functionality was not achieved[1].

We will briefly summarize the changes made within the Linux kernel to implement IEEE 802.11p, and follow with an overview of the Linux kernel modules developed to implement IEEE 1609.3/4, and the usage of these via a modified version of the userspace standard networking utility `iw` to enable transmission and reception of messages specified in the SAE J 2735 DSRC message dictionary [13]. Complete descriptions and source code are available in our public GitHub repository[2].

2.1 Modifications of `mac80211` subsystem

The following modifications were made to the `mac80211` subsystem found within `/net/wireless/mac80211`:

1. Setting the wildcard BSSID during configuration of the networking interface when the BSSID fetched does not match wildcard BSSID (FF:FF:FF:FF:FF),
2. Add a bit to list of hardware interface modes for specifying OCB mode
3. Check for the existence of concurrent network interfaces while configuring OCB mode and return an error if any are discovered, and
4. Replace usage of certain OCB functions defined in `ocb.c` with those defined in `ibss.c`.

2.2 Modifications of `cfg80211` utility

The following modifications were made to the `cfg80211` utility found at `/include/net/cfg80211`:

1. Add ability to define channels with 5/10MHz bandwidth, and
2. Include 5/10MHz channels in comparison statements evaluating and returning channel state information.

2.3 Modifications of `nl80211` subsystem

The following modifications were made to the `nl80211` utility defined at `/include/uapi/linux/nl0211.h` and `/net/wireless/nl80211.c`:

1. Supporting usage and configuration of OCB mode by networking interface,
2. Add definition of 5/10MHz-wide channels, and
3. Allowing usage of ITS channels exclusively in OCB mode.

2.4 Additional modifications of wireless subsystem

The remaining changes to the Linux kernel networking subsystem, found at `/net/wireless`, include the following:

1. Specification of OCB mode in network interface configuration,
2. Definition of functions for additional configurations when in OCB mode,
3. Support for the deinitialization of OCB mode by network interface,
4. Setting the network interface BSSID to the wildcard BSSID when configuring for OCB mode,
5. Allowing usage of ITS channels exclusively in OCB mode, and
6. Addition of support for 5/10MHz-wide channels.

[1] https://lwn.net/Articles/611635/
[2] https://github.com/p3n3troot0r/Mainline-80211P/

2.5 Modifications needed for wireless hardware drivers

Limited modification of a compatible wireless driver from within `/drivers/net/wireless` is necessary for all open-source COTS wireless hardware tested in this study. The initial driver chosen was `ath9k` and additional support was later added for `rtlwifi`. The ITS channels within the 5GHz radio band, when not beyond the physical limitations of the hardware, are beyond the range which most COTS wireless hardware is designed for. The following specific changes were made to the `ath9k` and `rtlwifi` drivers to implement IEEE 802.11p:

1. Definition of OCB mode as a networking mode,
2. Incorporation of OCB mode into list of hardware capabilities,
3. Definition of ITS-G5 channels in 5.9GHz band,
4. Definition of and support for 5/10MHz-wide channels, and
5. Enable user modification of the hardware regulatory domain.

2.6 WAVE Short Message Protocol

The WAVE Short Message Protocol (WSMP) is implemented as a kernel module that provides mechanisms for encoding and decoding WAVE message primitives. This includes the WSM, WSA, Service Info Segment, Channel Info Segment, WAVE Routing Advertisement, and Information Element Extension. Encoding/decoding is done with strict compliance to the standards. The module also includes a utility for handling the WAVE-specific p-encoding.

After constructing a WSM to transmit, the message is encoded using `wsmp_wsm_encode`. This returns a byte array that is passed along to the MAC/PHY layers.

2.7 Userspace tools for V2X stack

In order to use the V2X stack detailed herein with a standard Linux distribution, the CRDA and `wireless-regdb` utilities must be modified to allow specification and seamless transition to use of a custom regulatory domain. Modified versions of these utilities are available in the GitHub repository referenced previously.

3 Conclusions

We have presented our implementation of V2X through the IEEE 1609 standards. With this, we hope to engage and leverage the security community in the development of vehicular communications standards, and to facilitate growth and widespread interest in securing ITS infrastructure. The V2X stack we provide is licensed under the GNU General Public License v2 to promote collaborative development.

Example Usage

The following source code and output demonstrates the creation, encoding, and decoding of a WAVE Short Message:

Source Code

```
#include "common.h"
#include "encode.h"
#include "decode.h"
#include <time.h>
#include <stdlib.h>
#include <assert.h>

int main(int ac, char **av) {
  int err = 0;
  size_t *len;
```

```
    struct wsmp_wsm *msg = calloc(1, sizeof(struct wsmp_wsm));
    struct wsmp_wsm *parsed = calloc(1, sizeof(struct wsmp_wsm));

    msg->subtype = 0;
    msg->version = WSMP_VERSION;
    msg->use_n_iex = 1;
    msg->tpid = 0;

    msg->n_iex = calloc(1, sizeof(struct wsmp_iex));
    msg->n_iex->count = 3;
    msg->n_iex->chan = 172;
    msg->n_iex->data_rate = 3;
    msg->n_iex->tx_pow = 30;
    msg->n_iex->use[WSMP_EID_CHANNEL_NUMBER_80211] = 1;
    msg->n_iex->use[WSMP_EID_DATA_RATE_80211] = 1;
    msg->n_iex->use[WSMP_EID_TX_POWER_USED_80211] = 1;

    uint8_t tmp = 0;
    msg->psid = p_to_hex(0xC00305, &tmp);

    if (tmp != 3)
    goto out;

    msg->len = 13;
    msg->data = calloc(msg->len, 1);

    char str[] = "Hello world!";
    memcpy(msg->data, str, msg->len);

    print_wsm(msg);

    size_t count = 0;
    uint8_t *bytes = wsmp_wsm_encode(msg, &count, &err, WSMP_STRICT);

    if (err)
    goto out;

    printf("\nEncoded WSM (%lu bytes):\n", count);
    int i;
    for (i = 0; i < count; i++)
    printf("%02x ", bytes[i]);

    size_t parsed_index = 0;
    parsed = wsmp_wsm_decode(bytes, &parsed_index, count, &err, WSMP_STRICT);

    if (err)
    goto out;

    size_t parsed_count = 0;
    uint8_t *parsed_bytes = wsmp_wsm_encode(parsed, &parsed_count, &err, WSMP_STRICT);

    if (err)
    goto out;

    /* Equality check */
    for (i = 0; i < count; i++)
    if (bytes[i] != parsed_bytes[i])
    goto out;

    printf("\n\nRecovered Encoding (%lu bytes):\n", parsed_index);
    for (i = 0; i < parsed_count; i++)
    printf("%02x ", parsed_bytes[i]);

    printf("\n");

    out:
    free_wsm(msg);
    free_wsm(parsed);

    return err;
}
```

Output

```
BEGIN WSM
subtype:    0
version:    3
tpid:       0
use_n_iex: 1
n_iex:

BEGIN IEX
count:              3
chan:               172
data rate:          3
tx_pow:             30
psc.len:            0
psc.data:

ip:                 00000000000000000000000000000000
port:               0000
mac                 000000000000
rcpi_thres:         0
count_thres:        0
count_thres_int:    0
edca.ac_be:         00000000
edca.ac_bk:         00000000
edca.ac_vi:         00000000
edca.ac_vo:         00000000
chan_access:        0
repeat_rate:        0
loc_2d.latitude:    00000000
loc_2d.longitude:   00000000
loc_3d.latitude:    00000000
loc_3d.longitude:   00000000
loc_3d.elevation:   0000
advert_id.len:      0

sec_dns:            00000000000000000000000000000000
gateway_mac:        000000000000

raw_count:          0
raw:
in_use:
(0, 0)
(1, 0)
(2, 0)
(3, 0)
(4, 1)
(5, 0)
(6, 0)
(7, 0)
(8, 0)
(9, 0)
(10, 0)
(11, 0)
(12, 0)
(13, 0)
(14, 0)
(15, 1)
(16, 1)
(17, 0)
(18, 0)
(19, 0)
(20, 0)
(21, 0)
(22, 0)
(23, 0)

END IEX

psid:       00004385
ports.src:  8543
ports.dst:  0000
use_t_iex:  0
length:     13
data:
```

```
48 65 6c 6c 6f 20 77 6f 72 6c 64 21 00
END WSM

Encoded WSM (29 bytes):
0b 03 04 01 1e 0f 01 ac 10 01 03 00 c0 03 05 0d 48 65 6c 6c 6f 20 77 6f 72 6c 64 21 00

Recovered Encoding (29 bytes):
0b 03 04 01 1e 0f 01 ac 10 01 03 00 c0 03 05 0d 48 65 6c 6c 6f 20 77 6f 72 6c 64 21 00
```

References

[1] IEEE Std 802.11-2012 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

[2] IEEE Std 1609.0-2013 IEEE Guide for Wireless Access in Vehicular Environments (WAVE) – Architecture

[3] IEEE Std 1609.2-2016 IEEE Standard for Wireless Access in Vehicular Environments – Security Services for Applications and Management Messages

[4] IEEE Std 1609.3-2016 IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Networking Services

[5] IEEE Std 1609.4-2016 IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Multi-channel Operation

[6] IEEE Std 1609.11-2010 IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Over-the-Air Electronic Payment Data Exchange Protocol for Intelligent Transportation Systems (ITS)

[7] IEEE Std 1609.12-2016 IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Identifier Allocations

[8] ETSI EN 302 636-4-1 V1.2.1 (2014-05), Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality.

[9] ISO 29281-1, Intelligent transport systems — Communications access for land mobiles (CALM) — Non-IP networking—Part 1: Fast networking & transport layer protocol (FNTP).

[10] E. Donato, E. Madeira and L. Villas, "Impact of desynchronization problem in 1609.4/WAVE multi-channel operation," 2015 7th International Conference on New Technologies, Mobility and Security (NTMS), Paris, 2015, pp. 1-5.

[11] C. Valasek, C. Miller, "Remote Exploitation of an Unaltered Passenger Vehicle," 2015

[12] William Whyte, Jonathan Petit, Virendra Kumar, John Moring and Richard Roy, "Threat and Countermeasures Analysis for WAVE Service Advertisement," IEEE 18th International Conference on Intelligent Transportation Systems, 2015

[13] SAE J 2735-2016 Dedicated Short Range Communications (DSRC) Message Set Dictionary