The Security Testbed for the Purposes of the ITS-G5 Communication Attacks Prevention

Jiri Pokorny[1,2], Pavel Seda[1], Jan Dvorak[1], Lukas Malina[1], Zdenek Martinasek[1]
[1]Department of Telecommunications, Faculty of Electrical Engineering and Communication, Brno University of Technology
[2]Unit of Electrical Engineering, Tampere University, Korkeakoulunkatu 7, 337 20 Tampere, Finland
✉Contact author's e-mail: jiri.pokorny@vut.cz

***Abstract*—Secure communication in the Intelligent Transport System (ITS) plays a crucial role in vehicular safety. Security threats can be an unwanted cause of congestions and attacks. In this paper, first, the security threats in ITS are described and discussed. Second, a concept of the security testbed for ITS-G5 communication was presented. Its purpose is to test or verify the security threats for the machine-to-machine communication in the ITS. The testbed is composed of two parts. The first part represents the vehicle, and the second part is the Road-Side Unit (RSU) or the Road-vehicle unit (RVU). The testbed contains Arduino-type modules, SPI interface to CAN bus converter, and ELM 327 diagnostic tool supporting all communication protocols of the OBD standard. The simulator presented in this article was practically implemented and the functionality verified by experimental testing. Finally, a message for remote speed limiting was implemented on the testbed for further security testing.**

***Index Terms*—Intelligent transport system, ITS-G5, vehicle, simulator, OBD, CAN, Arduino.**

I. Introduction

Increasing vehicular traffic and pollution in urban areas was one of the motivators for designing the Intelligent Transport System (ITS) [1]. The goal of the ITS is to reduce traffic congestion by sharing the state of the surrounding traffic and events to other nearby vehicles. In the last decade, this system has been adopted by many different modes of transport.

The ITS is facing many security threats due to its open nature. Major accidents, loss of life, and property damage might be a consequence of security breaches. Various security properties can mitigate these threats. Standard properties defined by European Telecommunications Standards Institute (ETSI) are availability, integrity, authenticity, confidentiality, non-repudiation, and accountability [2]. The current solutions should be developed in order to provide the above requirements and prevent well-known security threats. More about the security of hybrid vehicular communication, including ITS-G5, is discussed in [3].

Currently, the ITS-G5 employs I2V (Infrastructure to Vehicle) or V2I (Vehicle to infrastructure) communications [4]. These two types of communications are most frequently sending warning or notification messages that are displayed in the car or at the computer of the infrastructure operator [5]. These messages usually include weather information [4], [5], service alerts [6] and many more use case. However, in use cases with serious warnings and notifications exchange, the security of the ITS-G5 communication should be a priority. For example, consider the use case of speed limiting the cars violating the rules. In this case, it is not only about notification, but the infrastructure or the integrated rescue system services needs to send an ITS-G5 message that is strongly secured by default.

This paper is organized as follows. Section II highlights the security concepts based on the PKI deployment and ITS-G5 attack classes. Section III describes OBD-II and other technical aspects of how the communication is integrated into the vehicles. Section IV describes the implemented security testbed for secured ITS-G5 communication using an OBD-II simulator implemented in Arduino. Finally, Section V concludes the work and summarizes the results from security analysis and the provided security testbed implementation.

A. Related Works

Nowadays, many scientific and research groups engage in modern technologies for wireless connection of vehicles, pedestrians, and infrastructure. The implementation of this communication often consists of communication modules with closed software, non-modular, unsecured, and supporting only one type of wireless communication, which can be disadvantageous and inappropriate. Several professional articles have dealt with this issue in recent years.

In [6], the authors present a study of several scenarios dealing with communication technology for the transmission of alerts and road information, or traffic information between vehicles. Cooperative Intelligent Transportation Systems (C-ITS) communication technology using the ITS-G5 protocol was used to transfer information. The system has been practically tested among vehicles in Finnish weather conditions. Their system has a minimum network latency and packet loss and meets the complete system requirements for generating alerts.

The authors in [4] compare the ITS-G5 communication technology with 5G test network (5GTN) technology. The authors used these technologies to transmit alerts and road information between vehicles and broadcasting stations in real-time. The article compares these technologies using User Datagram Protocol (UDP) or Transport layer Connection Protocol (TCP) transport protocols.

The article [5] deals with the integration of ITS-G5 technology with 5G technology to create an advanced heterogeneous testbed. The authors state that this testbed is used to create an architecture for Vehicular Ad hoc Networks (VANETs), providing real-time intelligent traffic services (road condition alerts and traffic information). The presented system was verified on a test track.

Finally, the authors in [7] present a new toll system solution using ITS-G5 technology. Secure communication meeting the basic security requirements is used to perform the toll transaction. The security of the proposed method is tested using the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool. The architecture proposed by the authors requires adequate resources that are suitable for V2I communication.

None of the mentioned articles deal with the security aspects of ITS-G5 communication between devices. In this paper, we investigate such needs using the provided security testbed focused on the security aspects of communication ITS-G5. Further, the ITS-G5 testbed verifies the aspects of communication and highlights the needs that have to be considered for the secured communication.

II. On Security in ITS

Emerging C-ITS safety applications depend on reliable and trustworthy data communication interfaces such as V2X, V2I, and V2V. The European Telecommunications Standards Institute (ETSI) currently releases standards that define a security framework for C-ITS, mainly based on using PKI (Public Key Infrastructure). This approach solely solves trust, data integrity and provides a certain level of user privacy by using pseudonyms in certificates. Regularly changing the pseudonym IDs in V2X communications can prevent user profiling and tracking. Nevertheless, data confidentiality is very limited in C-ITS schemes.

A. Deploying PKI in ITS-G5 Security

The ITS-G5 communication technology is mainly decentralized. Thus symmetric cryptography cannot be directly deployed to secure transmitted data. Further, the European Commission is working on a single pan-European trust model, which is based on the Public Key Infrastructure (PKI) [2] principle. The PKI hierarchy contains a certification authority (CA) that releases certificates with public keys that are then used to check digital signatures of the Cooperative Awareness Message (CAM), Decentralized Environmental Notification Message (DENM), or Infrastructure to Vehicle Information Message (IVIM) messages. Public keys *pseudonyms* are assigned by the Pseudonym Certificate Authority (PCA). Pseudonymous certificates are used to secure data integrity and authenticity and prevent tracking and profiling a specific vehicle and its users. A vehicle hidden behind a pseudonym can only be identified by a relevant authority that issued the vehicle pseudonym or by the Long Term Certification Authority (LTCA) authority that equips the vehicle by a long-term Long Term Certificate (LTC) certificate. These LTC certificates are used to authenticate with the PCA authority and are applied by car manufacturers. LTC is valid for the life of the car. Each vehicle has several pseudonyms with various validity periods. This approach prevents finding connections between pseudonyms and a specific vehicle. Nevertheless, the vehicle must be online and connected to the PCA backend server, assigning and changing pseudonyms.

On the top of the PKI hierarchy is the Root Certificate Authority (RCA). It ensures compliance with the rules and issues certificates to subordinate authorities (LTCA and PCA). Governments or private organizations can run these RCAs, but their number is very limited [8]. Nonetheless, complete (full) anonymity is against users' responsibility and general security. Therefore, the possibility of withdrawing the certificate and removing the vehicle from further communication must be maintained in some serious incidents.

B. ITS-G5 Attack Classes

ITS-G5 communication technology exchanges usually sensitive data. Messages such as CAM, DENM, IVIM or safety warning messages can be considered as the most sensitive and essential for secure their integrity and authenticity. CAM messages transmitted by a On-Board Unit (OBU) located in a moving vehicle may cause a non-intentional determination of the direction, speed, time, and location of the vehicle. An attacker can then easily track a specific vehicle (an user). Messages DENM or IVIM, which are transmitted by Road-Side Unit (RSU) or Road-Vehicle Unit (RVU) and serve for the protection of the driver from danger, can be easily misused, and the attacker can send those fake messages. For example, the vehicle can be forced to evade a maneuver that will cause a genuine accident. The attacker can also take complete control of the traffic or completely disables communication in ITS-G5 technology. Thus, the ITS-G5 communication technology can be exposed to various security threats and possible attacks that are categorized as follows [9]:

- Network Attacks – these attacks on ITS-G5 communication between RSU or RVU and OBU try to disturb or denial services. For example, these attacks can be (D)DoS attacks.
- Application Attacks – these attacks try to tamper or modify exchanged messages in ITS-G5 communication to cause damage, accidents, and inconvenience to valid users.
- Timing Attacks – attackers insert a command into the message (CAM, DENM, IVIM), causing a delay in the delivery of the message. The attacker does not violate the content of the message in any way but only causes that the message is not delivered at a required time. The most time-prone are security messages where delay (even minimal) may cause that messages lose their relevance, such as break alerts.
- Social Attacks – some malicious users may send offensive, vulgar messages to other vehicles. These messages can disturb the driver that may even cause a vehicle accident.
- Monitoring Attacks – attackers monitor traffic across the network and eavesdrops on communications between vehicles and the infrastructure. If an attacker discovers any helpful information, she/he can use it for his/her benefit or pass it on to another person. The potential

victim can be the police, who communicate through the network and plan interventions or operations. An attacker could eavesdrop on the entire communication and then warn specific parties [9].

III. On ITS-5G Communication Protocols and Interfaces

This section presents the standards for OBD and OBD-II that are used to connect diagnostic equipment and monitor or adjust electrical and electronic parts of the car. Further, an explanation of the core principles of communication between the electronic parts of the car via the CAN bus is provided, and the different communication protocols of the OBD-II standard are described.

A. OBD (On-Board Diagnostics)

On-Board Diagnostics (OBD) is a standard that was originally designed to diagnose and reduce emissions and to monitor the performance of major engine parts. It also allows external electronics to be linked to electronically controlled parts of the car. OBD monitors and detects faulty vehicle components that contribute to exhaust emissions, thereby reducing the release of pollutants into the air. The OBD also collects data from faulty components from various sensors to regulate them or to warn the driver of a fault. It stores the faults in memory so that they can then be displayed on one of the external diagnostic devices.

The basic diagnostic software OBD only allows interfacing with control units connected to emission systems. For connection to the airbag, navigation, or ABS control unit, special diagnostic software must be used, e.g., VCDS/VAG (Volkswagen/Audi/Skoda/Seat), BimCOM (BMW/Mini) [10].

1) OBD-I: One of the first standards was OBD-I. The purpose of the OBD-I standard was to encourage car manufacturers to design reliable and efficient emission control systems. A malfunction of a particular vehicle component was indicated by a flashing light, which most often had an engine symbol. When Data Link Connectoru (DLC) was connected, which was defined by each manufacturer, the light changed to a two-digit number from which the fault can be identified. Each manufacturer defined its diagnostic connector, but also the location of the connector or the procedure for identifying the fault, so there was no universal model.

2) OBD-II: The OBD-II standard was introduced because of the versatility and unification of vehicle diagnostics across car manufacturers. OBD-II specifies connector type, pin locations, signaling protocols, etc. Diagnostic Trouble Code (DTC) is a group of five-digit alphanumeric codes used to identify and diagnose a vehicle-related problem. Thanks to the OBD-II standard, connecting to any vehicle's onboard computer using one universal device is possible. OBD-II is used to monitor the emission control system and significant engine parts, among other things.

3) EOBD: It is a European modification of OBD-II. EOBD complies with EU emission regulations and does not differ from OBD-II [10].

B. OBD-II communication protocols

The OBD-II interface uses five protocols. Each protocol uses a different set of pins on the J1962 connector. The different protocols are described in the following paragraphs [10].

1) SAE J1850 PWM: Pulse-width modulation is used to transmit the signal of this protocol. The transmission rate is 41.6 kb/s. Pins 2 and 10 are used for communication, and the level *logic 1* is represented by +5 V. The word length is 12 b, and a cyclic checking mechanism is applied to check the correctness of the transmitted bits. It is used, for example, in Ford vehicles.

2) SAE J1850 VPW: The SAE J1850 VPW protocol uses variable pulse width for transmission. The transmission rate is 10.4 kb/s or 41.6 kb/s. Pin 2 is used for communication, and the level *logic 1* is represented by +7 V. The protocol is used by General Motors.

3) ISO 9141-2: This protocol uses asynchronous serial communication at 10.4 kb/s. The communication is similar to the RS232 standard, but the voltage levels are different. It uses pin 7 (K-line) and optionally pin 15 (L-line). Communication is bidirectional on a single wire (K-line) without a handshake (automated determination of parameters for communication before the actual data transfer). Signaling is done using a universal asynchronous receiver-transmitter (UART) interface that works with asynchronous serial communication and can be used to set the format and baud rate. The ISO 9141-2 protocol is used by European and Asian cars.

4) ISO 14230 KWP 2000: KWP2000 is a diagnostic protocol defined in the ISO 14230 standard. The ISO 14230 standard specifies the format of the transmitted data or the basic commands for communication. The KWP2000 protocol can also be used to update the firmware of the vehicle control unit.

5) ISO 15765-4/SAE J2480 (CAN): Bosch developed the CAN protocol for automotive and industrial control. Since 2008, all vehicles sold in the US have been required to implement CAN as one of their protocols. It uses pins 6 (Can High) and 14 (Can Low).

C. CAN bus

Controller Area Network (CAN) is a serial data bus developed by Bosch. The goal was to create a bus that would save cables and perform adequately in harsh environments, especially in industrial and automotive environments. With ever-increasing demands and the increasing number of electrical and electronic devices in the car, there was a need to design an efficient and reliable solution. The CAN data bus connects the various systems and sensors in the car.

The physical layer of the CAN bus is defined by ISO 11898-2 for high-speed CAN and ISO 11898-3 for low-speed CAN. For the high-speed CAN version, the transfer rate can be up to 1 Mb/s, for the low-speed CAN version up to 125 kb/s. However, the high-speed CAN version is currently the most commonly used. Data is transferred between individual systems and sensors over a maximum of one pair of wires. The pairs are designated as CAN-H and CAN-L.

CAN recognizes values as a recessive and dominant state. The recessive state is represented by *logic 1* and is defined such that there is the same potential on both CAN-H and CAN-L wires (CAN-H and CAN-L is 2.5 V). The dominant state is represented by *logic 0* and occurs when a potential difference of 2 V occurs (CAN-H is 3.5 V and CAN-L is 1.5 V). The transmission rate is guaranteed up to a distance of 40 m, with increasing distance, the rate decreases [11].

IV. Experimental Testbed

For purposes of security testing, we assembled a testbed that simulates a vehicle and a communication link between a vehicle and an RVU or RSU. The testbed consists of two parts which are described in this section, and Fig. 1 shows the block diagram.

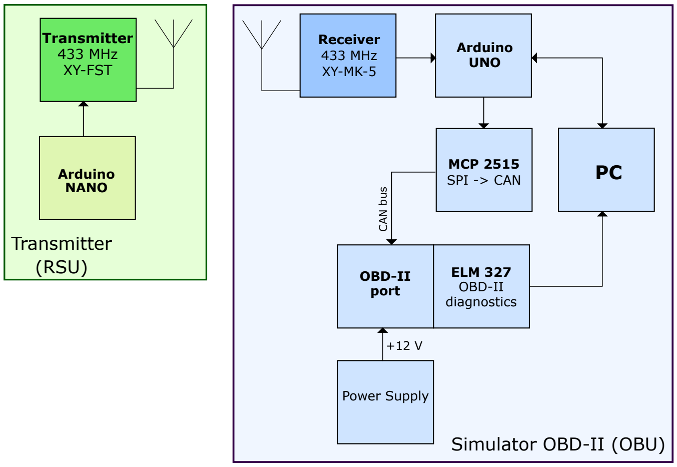


Fig. 1. Schematic of the end-to-end experimental testbed.

A. OBD-II Simulator

Each modern vehicle contains units, sensors, and controllers that communicate over CAN bus. Data on the CAN bus can be accessed via On-Board Diagnostics II (OBD-II) interface. The purpose of the OBD-II simulator is to simulate a vehicle in laboratory conditions. Such a simulator can serve testing purposes for security and new features testing. The schematic of our simulator is shown in Fig. 2, each component is described below.

- Arduino UNO: Program simulating a vehicle's Electronic Control Unit (ECU) is running on Arduino UNO. The program generates vehicle type data, e.g., vehicle speed, engine revolutions, throttle position, fuel tank level. For this simulator, the data transmitted by the ECU can be set from Arduino OBD 2 Simulator, originally developed by Khairnar and published on Github [12]. The graphical user interface is shown in Fig. 3.
- MCP 2515: This module is connected to the Arduino UNO and converts the SPI interface into CAN bus protocol. The data is further processed by diagnostic ELM327.

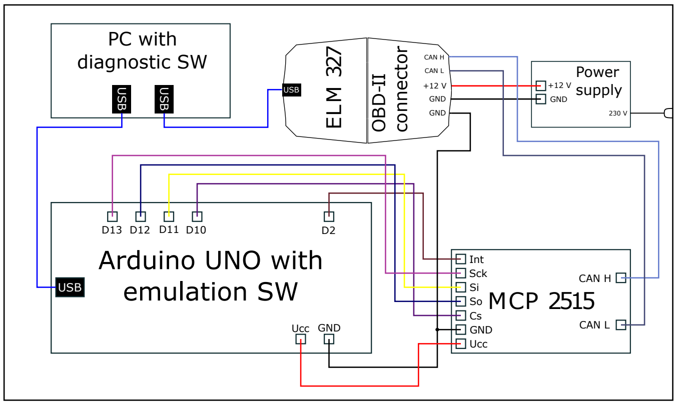
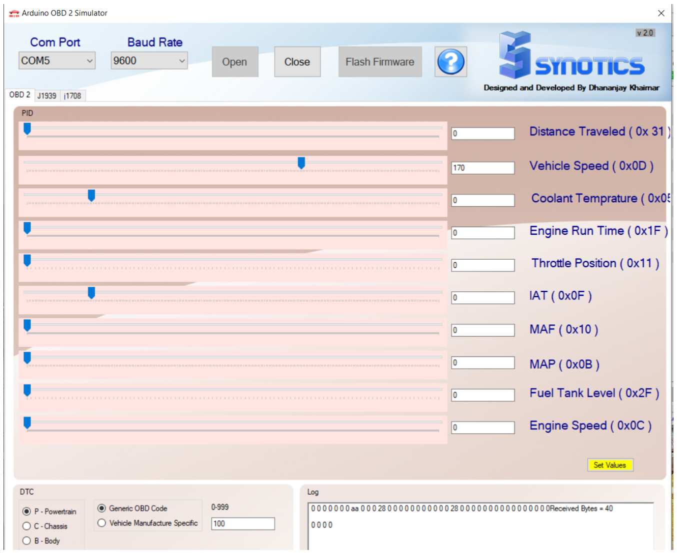


Fig. 2. Schematic of the OBD-II Simulator.

- ELM 327: Diagnostic ELM 327 supports all five communication protocols of the OBD standard, including ISO 15765-4 used for CAN bus. The diagnostic can read the data from the ECU and process them further. It communicates with a computer through a USB port where specialized software must be present to decode the data. One example of such software is the ScanMaster-ELM. This program has many functions. However, to verify the simulator's functionality, only reading and decoding of the received data is necessary. The diagnostic requires an input of 12 V for correct functionality and the two signal pins CAN-H and CAN-L for transfer of the data. The ground signal is connected to the ground of Arduino UNO.


Arduino OBD 2 Simulator

Com Port: COM5
Baud Rate: 9600

PID	
0	Distance Traveled (0x 31)
170	Vehicle Speed (0x0D)
0	Coolant Temprature (0x05
0	Engine Run Time (0x1F)
0	Throttle Position (0x11)
0	IAT (0x0F)
0	MAF (0x10)
0	MAP (0x0B)
0	Fuel Tank Level (0x2F)
0	Engine Speed (0x0C)

DTC
● P - Powertrain
○ C - Chassis
○ B - Body
● Generic OBD Code
○ Vehicle Manufacture Specific
0-999
100

Log
```
0 0 0 0 0 0 0 aa 0 0 0 28 0 0 0 0 0 0 0 0 0 0 0 0 28 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0Received Bytes = 40
0 0 0 0
```

Fig. 3. Desktop application for Arduino OBD-II Simulator.

After interconnecting all parts, as shown in Fig. 4, a link between diagnostic software and the diagnostic must be set up. Connection is made automatically, the program first identifies the ELM327 diagnostic, which then sends a control frame

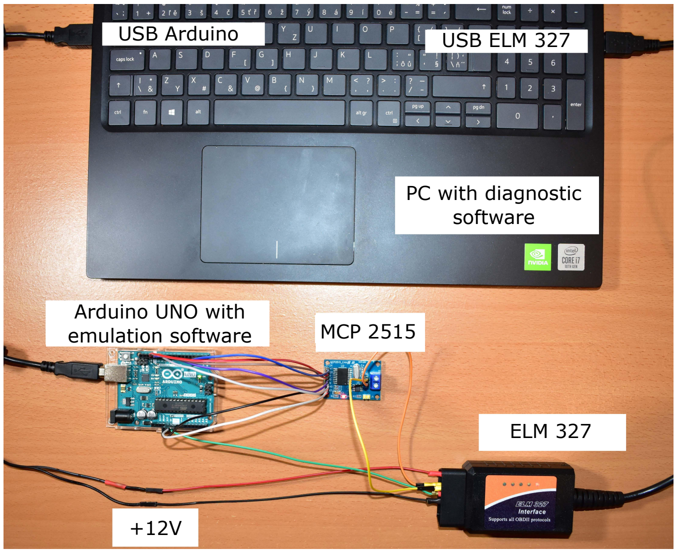

USB Arduino
USB ELM 327
PC with diagnostic software
Arduino UNO with emulation software
MCP 2515
ELM 327
+12V

Fig. 4. Real interconnection of OBD-II Simulator.

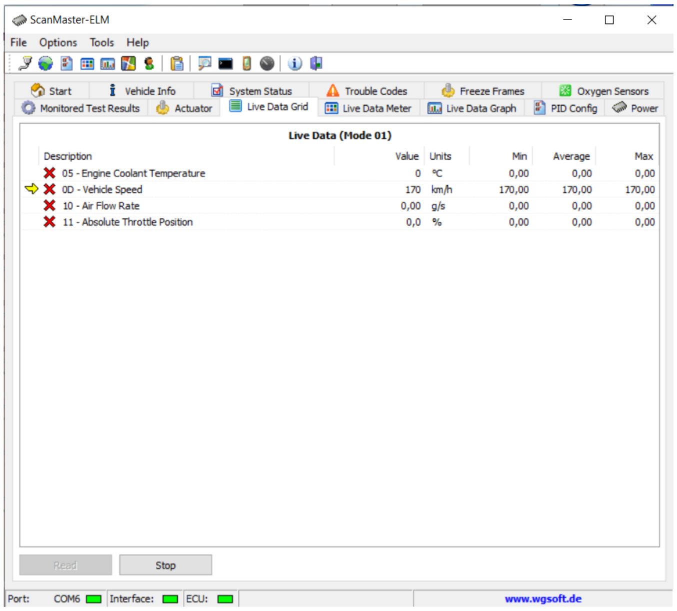
to Arduino UNO. If the frame is received correctly, the connection is set up, and the software prints out which protocol is used for communication. In our case, it is the ISO 15765-4. As soon as the connection is established, it is possible to read the data from the simulated ECU.

Parameters must first be manually selected in the software to achieve a reading of the values. The parameters are selected based on their PIDs. PIDs are two-digit hexadecimal values where each value represents one parameter of the vehicle. For demonstration purposes, the parameter **0D** was selected. This parameter represents the vehicle speed. Fig. 3 shows the vehicle speed parameter was set to 170 km/h. Arduino UNO processes this value and sends it to the diagnostic. The diagnostic software reads the values and prints them on the screen, as is shown in Fig. 5.

B. Communication link between a vehicle and an RVU/RSU

The communication between entities in ITS-G5 infrastructure runs on the IEEE 802.11p standard. This standard is not so widespread, and obtaining devices supporting it was not possible at this moment. For testing purposes, the IEEE 802.11p wireless links can be substituted with other wireless technology. The choice for our simulator was a technology working in an unlicensed spectrum. The selected set was the transmitter XY-FST with the receiver XY-MK-5V. The range of the devices can be up to 200 m with the transmitting speed of up to 10 Kb/s. When doing measurements, the drawback of the shared spectrum has to be taken into account. The schematic of the communication link is shown in Fig. 6.

The interconnection of the wireless link is shown in Fig. 7. Both the transmitter and the receiver are controlled externally. In our simulator, each device is connected to an Arduino board. Arduino UNO controls the transmitter, and Arduino NANO controls the receiver. The controlling code of the transmitter and receiver was based on the RadioHead library [13].



Live Data (Mode 01)

Description	Value	Units	Min	Average	Max
05 - Engine Coolant Temperature	0	°C	0,00	0,00	0,00
0D - Vehicle Speed	170	km/h	170,00	170,00	170,00
10 - Air Flow Rate	0,00	g/s	0,00	0,00	0,00
11 - Absolute Throttle Position	0,0	%	0,00	0,00	0,00

Port: COM6 🟩 Interface: 🟩 ECU: 🟩 www.wgsoft.de

Fig. 5. Reading data from the OBD-II Simulator with the ScanMaster-ELM diagnostic tool.

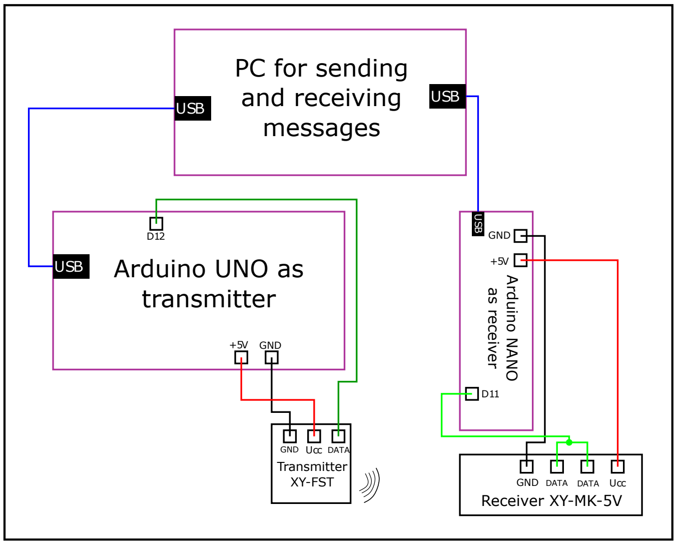


Fig. 6. Schematic of the transmitter and receiver.

C. Example message and simulation results

Currently discussed topic in vehicular safety and traffic efficiency is speed limiting [14]. Remote speed limiting managed by ITS messages from other vehicles or the infrastructure can prevent traffic congestion and increase road safety. Additionally, the remote speed limiting can be used in more critical cases, such as police chases. When the police chase a driver, the police would be able to slow down or stop the vehicle remotely. The procedure of remotely stopping the vehicle will require a high level of security.

For this purpose, we implemented an example message for changing the speed of a vehicle. The transmitter sends

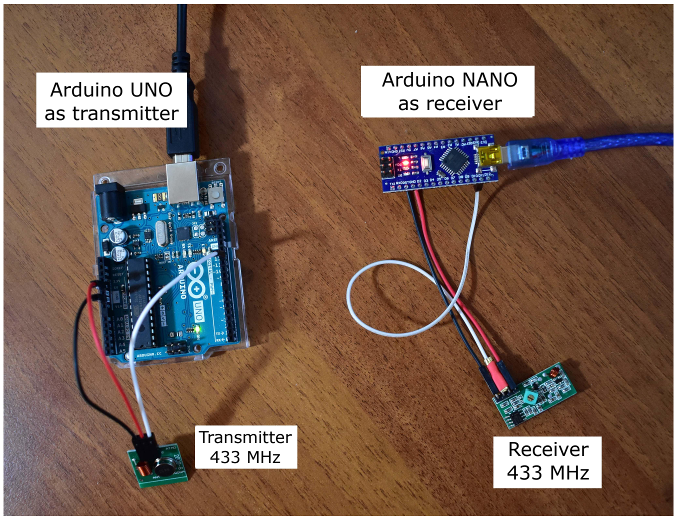


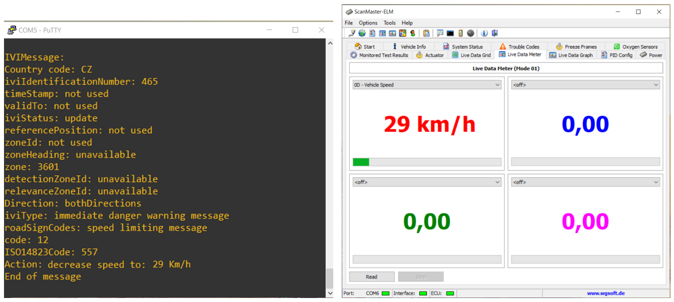
Fig. 7. Real interconnection of transmitter and receiver with Arduino controllers.

a message with the command to change the vehicle's speed to 29 km/h. The original speed of the vehicle was 170 km/h as was previously shown in Fig. 5. Fig. 8 shows the log from the speed limiting message from the transmitter as well as the vehicle's actual speed read from the OBD-II Simulator.

Further, an attack was simulated on the testbed. This scenario consists of one authentic RSU, one OBU, and one rogue RSU representing an attacker. The authentic RSU transmits a message every two seconds which makes a total of 30 messages per minute. The rogue RSU transmits messages in specific intervals. The number of successfully received messages per different interval was averaged over ten minutes, i.e., ten blocks of 30 messages. The plot showing the resulting values is in Fig. 9.

According to the results, the attacker is able to completely block the transmission of other RSUs when the interval between messages is lower than 100 ms. On the other side, no messages were lost when the attacker transmitted a message every six seconds. The size of the attacker's message had no effect on the number of the received authentic messages. This happens because the receiver receives messages from all sources and if the attacker's messages arrive before the authentic message, it prevents the receiver to receive it, since the receiver can only process one message at a time. These results provide a starting point for us to improve the probability of the number of successfully received messages.

Additionally, the transmitting signal was captured by a wireless probe. Fig. 10 shows three views, the authentic periodically transmitting RSU in time on the top, the detail of the transmitted message in time in the middle, and frequency spectrum of the transmitted signal on the bottom. Further the analysis of these signals can be helpful for improving the number of received messages by distinguishing the attacker from the authentic transmitter.



```
IVIMessage:
Country code: CZ
iviIdentificationNumber: 465
timeStamp: not used
validTo: not used
iviStatus: update
referencePosition: not used
zoneId: not used
zoneHeading: unavailable
zone: 3601
detectionZoneId: unavailable
relevanceZoneId: unavailable
Direction: bothDirections
iviType: immediate danger warning message
roadSignCodes: speed limiting message
code: 12
ISO14823Code: 557
Action: decrease speed to: 29 Km/h
End of message
```

0D - Vehicle Speed
29 km/h
<off>
0,00
<off>
0,00
<off>
0,00
Port: COM6 Interface: ECU:

Fig. 8. Implemented message with the goal of reducing a vehicle's speed with the verification of received data.

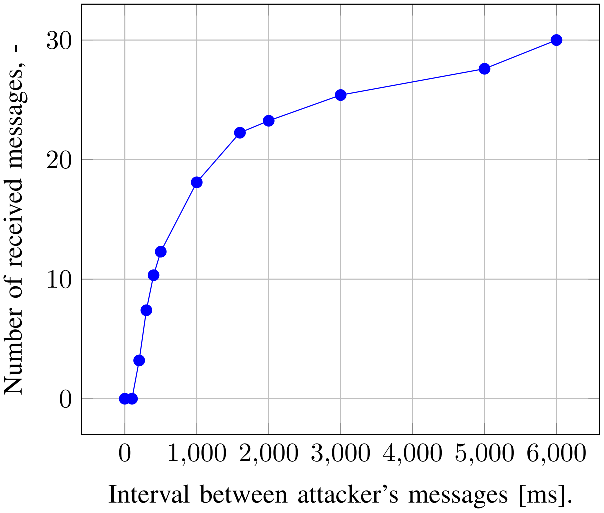


Fig. 9. Impact of interval between attacker's messages on the number of successfully received authentic messages.

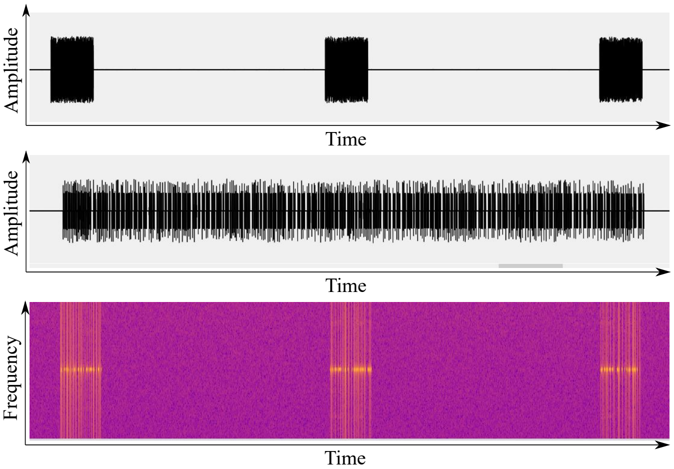


Fig. 10. Periodic authentic transmission, single transmitted message, and frequency spectrum.

V. Conclusion

This paper presented the current most crucial security threats of vehicular technology with a focus on ITS. Security violation in ITS might be sources for causing property damage and possible life loss. These were the main motivation factors for developing the presented security testbed.

The OBD-II Simulator was implemented on the Arduino platform, as well as the communication between OBU and the RSU/RVU by using the 433 MHz frequency from the ISM radio band. The functionality was verified with an example message for speed limiting. Also, an attack on the OBU was simulated and the impact of a interval between transmitted messages from the attacker was plotted. These results will be helpful for evaluation when further security techniques will be implemented. The testbed proved to be a good starting point for future research in security testing and threat prevention.

The correct function can be further tested with a real vehicle dashboard obtained from the damaged vehicle as future improvements. That would verify the function on the latest vehicle models which use the latest communication protocols. The testbed can also be improved by replacing the transmitter and receiver with wireless equipment used in the ITS-G5. Those are wireless devices supporting the IEEE 802.11p protocol. There is a minimal number of such devices on the market. However, devices with the same or similar chips can be purchased but require modifications of their firmware.

Acknowledgment

Research described in this paper was financed by the Ministry of Interior under grant VJ01010066.

References

[1] ETSI, "Automotive Intelligent Transport Systems (ITS)," [online]. [Acc. 2021-9-1]. [Online]. Available: https://www.etsi.org/technologies/automotive-intelligent-transport

[2] ——, "TR 102 893 v1.2.1 - Intelligent Transport Systems (ITS); Security; Threat, Vulnerability and Risk Analysis (TVRA)," [online]. [Acc. 2021-9-1]. [Online]. Available: https://www.etsi.org/deliver/etsi_tr/102800_102899/102893/01.02.01_60/tr_102893v010201p.pdf

[3] N. Bissmeyer, J.-F. van Dam, C. Zimmermann, and K. Eckert, "Security in hybrid vehicular communication based on its-g5, lte-v, and mobile edge computing," in *AmE 2018-Automotive meets Electronics; 9th GMM-Symposium*. VDE, 2018, pp. 1–6.

[4] M. N. Tahir, T. Sukuvaara, and M. Katz, "Vehicular networking: ITS-G5 vs 5G performance evaluation using road weather information," in *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 2020, pp. 1–6.

[5] M. N. Tahir and M. Katz, "Heterogeneous (ITS-G5 and 5G) vehicular pilot road weather service platform in a realistic operational environment," *Sensors*, vol. 21, no. 5, p. 1676, 2021.

[6] M. N. Tahir, K. Mäenpää, T. Sukuvaara, and P. Leviäkangas, "Deployment and Analysis of Cooperative Intelligent Transport System Pilot Service Alerts in Real Environment," *IEEE Open Journal of Intelligent Transportation Systems*, 2021.

[7] M. Randriamasy, A. Cabani, H. Chafouk, and G. Fremont, "Formally Validated of Novel Tolling Service With the ITS-G5," *IEEE Access*, vol. 7, pp. 41 133–41 144, 2019.

[8] Joint Research Centre, "Cryptographic security mechanisms of the next generation digital tachograph system and future considerations," [online]. [Acc. 2021-2-12]. [Online]. Available: https://publications.jrc.ec.europa.eu/repository/bitstream/JRC77933/lbna25663enn.pdf

[9] I. A. Sumra, I. Ahmad, H. Hasbullah, and J. bin Ab Manan, "Classes of attacks in VANET," in *2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC)*, 2011, pp. 1–5.

[10] K. McCord, *Automotive Diagnostic Systems: Understanding OBD I and OBD II*. CarTech Inc, 2011.

[11] A. A. Salunkhe, P. P. Kamble, and R. Jadhav, "Design and implementation of can bus protocol for monitoring vehicle parameters," in *IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, 2016, pp. 301–304.

[12] D. Khairnar, "Arduino OBD2 Simulator," [online]. [Acc. 2020-12-11]. [Online]. Available: https://github.com/8-DK/Arduino_OBD2_Simulator

[13] "How 433MHz RF Tx-Rx Modules Work & Interface with Arduino," [online]. [Acc. 2020-12-06]. [Online]. Available: https://lastminuteengineers.com/433mhz-rf-wireless-arduino-tutorial/#radiohead-library-a-swiss-army-knife-for-wireless-modules

[14] L. Bieker, "How Does the Traffic Behavior Change by Using In-Vehicle Signage for Speed Limits in Urban Areas?" in *Simulating Urban Traffic Scenarios*. Springer, 2019, pp. 37–46.