An experimental analysis of ECQV implicit certificates performance in VANETs

Francesco Pollicino, Dario Stabili, Luca Ferretti and Mirco Marchetti
University of Modena and Reggio Emilia, Italy
Email: {francesco.pollicino, dario.stabili, luca.ferretti, mirco.marchetti}@unimore.it

***Abstract*—Emerging Cooperative Intelligent Transportation Systems (C-ITS) enable improved driving experience and safety guarantees, but require secure Vehicular Ad-hoc NETworks (VANETs) that must comply to strict performance constraints. Specialized standards have been defined to these aims, such as the IEEE 1609.2 that uses network-efficient cryptographic protocols to reduce communication latencies. The reduced latencies are achieved through a combination of the Elliptic Curve Qu-Vantstone (ECQV) implicit certificate scheme and the Elliptic Curve Digital Signature Algorithm (ECDSA), to guarantee data integrity and authenticity. However, literature lacks implementations and evaluations for vehicular systems. In this paper, we consider the IEEE 1609.2 standard for secure VANETs and investigate the feasibility of ECQV and ECDSA schemes when deployed in C-ITSs. We propose a prototype implementation of the standard ECQV scheme to evaluate its performance on automotive-grade hardware. To the best of our knowledge, this is the first open implementation of the scheme for constrained devices that are characterized by low computational power and low memory. We evaluate its performance against C-ITS communication latency constraints and show that, although even highly constrained devices can support the standard, complying with stricter requirements demands for higher computational resources.**

I. Introduction and Related Work

The increasing popularity of connected vehicles allows to enable Vehicular Ad-hoc NETworks (VANETs) [1], that is one of the intelligent components forming the Cooperative Intelligent Transportation Systems (C-ITS) [2] with novel services and features that can improve driving experience through the cooperation between road-infrastructure services, road users and vehicles. However, designing and implementing this complex system is a challenging task. As an example, the novel communication networks must support a highly heterogeneous environment, that comprises many vehicles and boards manufacturers, and comply with the strict C-ITS constraints in terms of small latency and dynamic configuration of the network. In this paper, we investigate security solutions for VANETs and focus on integrity and authenticity guarantees of vehicles communications. To this aim, we consider the security protocols of the IEEE 1609.2 standard, that defines secure message formats for Wireless Access in Vehicular Environments (WAVE), policies for the management of the security certificates and the supported digital signature and encryption algorithms. Although the IEEE 1609.2 standard represents a comprehensive proposal for secure vehicle communications, it does not provide recommendations that take into account the potentially different characteristics of the VANETs in multiple scenarios. As an example, it does not consider how security solutions may vary depending on the number of connected devices or the constraints in terms of end-to-end communication latencies.

The paper includes two main contributions. First, we propose an implementation of the Elliptic Curve Qu-Vanstone (ECQV) implicit certificate scheme and the Elliptic Curve Digital Signature Algorithm (ECDSA) that is compliant to the IEEE 1609.2 standard, and describe an experimental evaluation of the schemes when deployed on automotive-grade boards [3]. To the best of our knowledge, this is the first open implementation of implicit certificates for resource-constrained devices in terms of computational power and memory. Second, we investigate the feasibility of the schemes in multiple VANET scenarios characterized by different latency constraints discussed by the National Highway Traffic Safety Administration (NHTSA) for vehicles safety [4]. Guaranteeing security of automotive networks requires protecting connected vehicles from cyber-attacks. Security proposals for intra-vehicular communications focus on defending against denial-of-service and message injection attacks by using special-purpose anomaly detectors [5]–[7] and network analysis tools [8], [9], and on protecting the integrity and authenticity of the ECUs messages by using lightweight cryptographic schemes and architectures [10], [11]. The implicit certificate schemes considered by this paper could also be of interest for protecting the integrity of high-tier and future intra-vehicular networks that can support asymmetric cryptography schemes. However, in this paper we focus on inter-vehicular networks, where their adoption on existing systems seems more promising. We leave further analyses on intra-vehicular networks as a future work.

The existing proposals [12]–[14] for vehicle-to-vehicle (V2V) communications that relate most with ours proposal are architectures for guaranteeing communications integrity and authenticity, that require the adoption of proper cryptographic protocols for identity and key management. They differ from standard Web protocols because they are designed to comply to the particular requirements of vehicles communications, including privacy protection against vehicles tracking and vehicle safety requirements. Although they also build over the IEEE 1609.2 standard, they do not consider the stricter workload requirements that characterize realistic vehicles communications (e.g., only consider messages sent every 100ms). Moreover, they only consider using standard X.509 certifi-

cates. In this paper, we analyze multiple stricter requirements in terms of allowed latency and throughput, and analyze the behavior of implicit certificates by using a prototype implementation. Similar to our proposal, the work presented in [15] analyzed if VANETs communications security based on ECQV certificates can satisfy latency constraints that are due to enforce safety in a crash incident. Although their analyses are based on multiple standards (IEEE 1609.2 [16], ETSI ITS-G5 [17] and ARIB STD-T109 [18]), the proposed experimental evaluation is based on a laptop computer and does not consider the timings of a proper implementation on hardware that typically characterizes automotive boards. The evaluation proposed in this paper is focused on realistic automotive-grade boards and is based on the first open prototype implementation for ECQV and ECDSA. The prototype implementation is optimized for low-power devices and allows to better analyze the performance of the implemented protocols in real-world scenarios.

The rest of the paper is organized as follows. We present base knowledge on the IEEE 1609.2 standard, and on ECQV implicit certificate and ECDSA schemes in Section II. We describe the main design choices of the proposed implementation and micro-benchmarks on automotive-grade boards in Section III. We investigate the feasibility of the schemes in VANETs in Section IV. Finally, we conclude the work in Section V.

II. Base knowledge on IEEE 1609.2 and ECQV

The IEEE 1609.2 [16] standard specifies the recommendations for Wireless Access in Vehicular Environments (WAVE), including the framework of the security services that all the devices must support, the formats used to exchange messages, and the cryptographic schemes and algorithms that can be used to protect messages. In this paper, we focus on the protocols used to guarantee integrity and authenticity of messages, that include digital signatures and certificates.

The reference standard for digital signatures is the Elliptic Curve Digital Signature Algorithm (ECDSA), that is a well-known NIST standard already used to protect communications in the Web. The IEEE 1609.2 standard allows the adoption of ECDSA with three elliptic curves identified as *NIST-P256*, *BrainpoolP256r1* and *BrainpoolP384r1*.

To build a scalable architecture for public key distribution, the IEEE 1609.2 standard requires deploying a Public Key Infrastructure (PKI) that is similar to that of the Web, where trusted *Certification Authorities* (CA) bind the identity information of communicating parties to their public key material within *certificates*. To this aim, the standard supports two types of certificates: traditional *certificate chains*, that are well-known and already used for Web communications, and *implicit certificates* [19], a less known alternative that allows to build smaller certificates.

Traditional certificates bind the identity information and the public key of a user by requiring CAs to produce a digital signature of the whole data. Certificate chains extend these certificates by allowing the *root certification authority* to delegate intermediate CAs to release certificate, called *intermediate CAs*, that can in turn delegate other CAs. Each delegation demands the delegating CA to sign a special-purpose certificate, including the identity and the public key of the delegated CA. Moreover, a certificate chain includes the public keys of the authorized user and of all the intermediate CAs (for this reason it could also be called an *explicit certificate*, in opposition to *implicit certificates*). For these reasons, the size of the certificate chain is proportional to the number of the intermediate CAs. Verifying the certificate chain requires to verify the digital signature attached by each CA up to the Root Certificate.

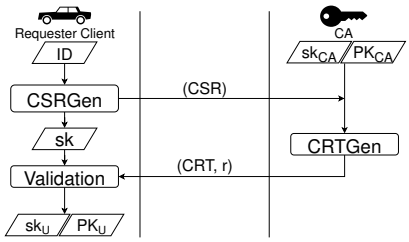

Fig. 1. Operation flow for the the generation of the ECQV certificate

The implicit certificate scheme uses a more complex approach that leverages particular mathematical properties to bind identity information and public key *without explicitly* storing them. The most popular protocol for implicit certificates, that is also included in the IEEE 1609.2 standard, is ECQV. Intuitively, an ECQV certificate does not include the public key of the sender, but allows a recipient to recompute it by using the certificate of the sender and the public key of the CA, thus saving network usage. Although implicit certificates seem very convenient thanks to their space efficiency, they have a few disadvantages that limit their adoption in common Web communications. Network savings of implicit certificates can be considered negligible in most Web scenarios, because communications are mostly operated through channels exchanging large amounts of data and certificates are only used once during the secure channels handshakes. However, vehicular networks distinguish from traditional Web communications because they are characterized by datagram-oriented communications that include small data (especially safety-critical packets, see Section III). Moreover, vehicular networks are deployed on possibly low-rate wireless networks and have tighter latency requirements. As a result, network savings of implicit certificates might be worth the more complex management procedure and reduced flexibility.

The ECQV operations framework includes four routines: *certificate sign request*, *certificate generation*, *certificate validation*, and *public key extraction*. These operations, combined with the *signature* and *verification* procedures of ECDSA, allow to guarantee message integrity and authenticity. In the following we describe the flow of the operations from the generation of an ECQV certificate to the signature verification by referring to Figures 1 and 2.

A *requester* is a client that requests a valid certificate to the

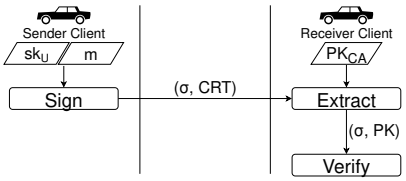

Fig. 2. Operations flow for the ECDSA signature based on ECQV certificates

CA by generating a *Certificate Sign Request* (*CSR*) via the *CSR generation* function (*CSRGen*). The *CSRGen* function requires the *ID* of the entity requesting the certificate, and produces an output composed by the *CSR* (that includes *ID* and *PK*, which is the intermediate public key of the requester) and the intermediate private key *sk*. The *CSR* is sent to the CA and, upon identity verification, produces the corresponding certificate *CRT* and a private key contribution *r* by using the *CRT generation* function (*CRTGen*). The *CRTGen* function requires the *CSR* and the key pair of the CA sk_{CA}, PK_{CA} to produce the *CRT*. After the reception of the *CRT* from the CA, the requester validates the certificate with the *CRT Validation* function (*Validate*). Upon verification, the requester generates the final key pair (sk_U, PK_U) by using *CRT* and *r*. The private key sk_U is used to generate the signature σ of the messages with the ECDSA *signing* function (*Sign*). The client uses the private key sk_U with the the ECDSA *signing* function (*Sign*) to sign messages. After the signature σ is computed, the client sends the message m, the signature σ, and its own certificate *CRT* by broadcasting them to the nearby clients. Upon reception of a message, each client uses the *extract* function (*Extract*) to extract the public key of the sender client from the *CRT*, which is later used for the verification of the signature of the message with the *verify* function (*Verify*). A summary of the used notation is presented in Table I.

TABLE I
SUMMARY OF SYMBOLS USED IN PROTOCOL

Symbol	Description
CA	Certification Authority
CSR	Certificate Sign Request
CSRGen	CSR generation function
CRT	Certificate
CRTGen	CRT generation function
Validate	CRT Validation function
Sign	Signature function
Extract	Public key extraction function
Verify	Signature verification function
PK	Public Key
sk	Secret key
sk_{CA}, PK_{CA}	Secret and Public key of the CA
sk_U, PK_U	Secret and Public key of the requester
r	Private key contribution
σ	Signature
m	Message

III. PROTOTYPE IMPLEMENTATION AND MICRO-BENCHMARKS

We describe the main design choices of the proposed implementation for ECQV implicit certificates and timings on a few representative platforms. The full implementation is open and available online [3]. The implementation supports the NIST P256 curve (*secp256r1*), that is one of the curves specified by the IEEE 1609.2 standard, and depends on the *microECC* [20] library for efficiently computing elliptic curve operations even on constrained devices thanks to optimized ASM code for ARM platforms. Although the IEEE 1609.2 standard also specifies two additional curves (*Brainpool256r1* and *Brainpool384k1*), each device is only required to support at least one of them. Moreover, the *secp256r1* curve is the most popular curve and the most suitable for constrained platforms, thus it seems the best choice for the proposed evaluation.

We observe that our implementation also supports four additional curves included in NIST standards for Web communications [21]: *secp160r1*, *secp192r1*, *secp224r1* and *secp256k1*. The curves *secp160r1*, *secp192r1* *secp224r1* grant better performance but guarantee lower security levels. In particular, *secp160r1*, *secp192r1* guarantee about 80-, 96-*bits* security and thus are considered deprecated for modern secure communications. Curve *secp224r1* guarantees 112-*bits* security and is considered secure until 2030, thus preventing it to be suitable for devices with $10+$ years of estimated lifetime. The *secp256k1* curve offers the same security level of *secp256r1* and better performance. For these reasons, in this paper we focus on both *secp256r1* and *secp256k1* curves. We remark however that only the *secp256r1* curve is supported by the IEEE 1609.2 standard, thus the *secp256k1* curve is included only for comparison.

The implementation complies to the following design choices and best practices:

- critical cryptographic operations do not use conditional branches and adopt time-constant code to avoid timing side-channels (e.g., scalar point multiplication [22]);
- no variables are allocated by using dynamic memory allocation;
- random numbers are generated with an hardware TRNG when available, otherwise we use software pseudo-random number generator functions;
- although the implementation supports multiple elliptic curves, the code selects only the required curve at compile time to avoid wasting storage by including useless code. The current version limits the selection of a single curve;
- elliptic curve points are always transferred by using a compressed representation to reduce the network overhead by using the format defined in the standard SEC 1 [23], while cryptographic operations are computed by using the uncompressed affine representation (x, y).

We present the timings required by the proposed implementation to compute ECQV and ECDSA operations on low-power ARM devices that are representative of different automotive-grade boards:

- RaspberryPi 4 Model B (**RPi4**): equipped with a Broadcom BCM2711 SoC based on 64-bit ARM Cortex-A72 quad-core CPU operating at 1.5GHz.
- RaspberryPi 3 Model B (**RPi3**): equipped with a Broad-

TABLE II
TIMINGS OF THE OPERATIONS ON THE *secp256r1* CURVE [ms]

	x86_64	**RPi4**	**RPi3**	**RPi1**	**M4**	**M3**
KeyGen	0.32	1.33	2.82	14.67	109.67	147
CsrGen	0.33	1.37	2.79	14.21	109.73	148
CrtGen	0.62	2.85	5.83	30.02	233.45	317
Validate	0.60	2.79	5.62	29.29	231.47	313
Extract	0.34	1.44	3.03	15.38	120.13	162
Sign	0.32	1.44	3.05	15.38	118.00	161
Verify	0.35	1.58	3.36	16.57	133.47	182

TABLE III
TIMINGS OF THE OPERATIONS ON THE *secp256k1* CURVE [ms]

	x86_64	**RPi4**	**RPi3**	**RPi1**	**M4**	**M3**
KeyGen	0.22	1.05	2.24	12.07	85.02	132
CsrGen	0.22	1.13	2.29	12.16	84.98	133
CrtGen	0.48	2.35	5.25	26.15	186.20	291
Validate	0.46	2.26	4.73	25.72	184.27	287
Extract	0.24	1.19	2.55	13.63	97.68	152
Sign	0.25	1.21	2.53	13.15	93.31	144
Verify	0.24	1.13	2.25	13.53	97.97	152

com BCM2837 SoC based on 64-bit ARM Cortex-A53 quad-core CPU operating at 1.2GHz.

- RaspberryPi 1 Model B (**RPi1**): equipped with a Broadcom BCM2835 SoC based on ARMv6 single-core CPU operating at 700MHz.
- STM32L4 (**M4**): equipped with 32-bit Cortex M4 CPU operating at 80MHz, with 1MB of flash memory and 128KB of RAM memory.
- Arduino Due (**M3**): equipped with an Atmel SAM3X8E, 32-bit ARM Cortex-M3 CPU operating at 84MHz, with 512KB of flash memory and 96KB of RAM memory.

Finally, we also include results of an **x86_64** architecture for completeness. The considered system is a modern laptop equipped with an Intel Core i7-9750H.

We observe that the implementation is single-threaded and does not take advantage of multi-core architectures. This is a typical design choice for most cryptographic algorithms, that cannot be easily parallelized without risking to introduce security vulnerabilities. We do not consider it as a limitation because low-power embedded devices are based on single-core architectures, and more powerful architectures can operate concurrent operations on multiple data.

Tables II and III show timing results of the proposed implementation by using the *secp256r1* and *secp256k1* curves respectively. The columns of the tables represent the platforms used for the evaluation, and the rows represent the considered cryptographic operations, namely (top to bottom), *key generation (KeyGen)*, *certificate request generation (CsrGen)*, *certificate generation (CrtGen)*, *validation (Validate)*, *extraction (Extract)*, *signature (Sign)* and *verification (Verify)*. All results are expressed in milliseconds. We observe that for each architecture, the timings of all the main ECQV operations are double the time of the ECDSA operations. Moreover, the most powerful ARM architectures (RPi4 and RPi3) can operate all operations within a few milliseconds, and cheaper ARM architectures (RPi1) are an order of magnitude slower. Finally, ultra-low power architectures (M4 and M3) are two orders of magnitude slower, requiring even a few hundreds of ms to operate each operation. Among all operations, we observe that although the certificate generation operation (*CrtGen*) can usually be deployed on a dedicated server machine without resource constraints, the implemented library would also be able to generate novel certificates within low-power devices.

However, we remark that future generations of autonomous vehicles might be equipped with dedicated devices with improved performance. The performance results of the implementation on the **x86_64** architecture should be the baseline for any future generation dedicated devices. We propose further analyses in the next Section IV, where we consider the requirements of a real vehicular communication.

IV. EXPERIMENTAL EVALUATION

We propose an experimental evaluation based on the *secp256r1* and *secp256k1* curves to analyze the feasibility of ECQV implicit certificates on embedded devices for securing vehicles communications. To this aim, we consider communication requirements as described by the vehicle safety communications project report of the National Highway Traffic Safety Administration (NHTSA) [24]. The report considers multiple vehicle communication scenarios and defines the constraints that must be satisfied to guarantee the safety of the vehicles and of the people within. Among these constrains, we are interested in the *allowable latency*, that is the total maximum latency that can be allowed for end-to-end communication and processing of information. The report identifies 5 different categories of safety-related application scenarios:

- Intersection Collision Avoidance;
- Public Safety;
- Sign Extension;
- Vehicle Diagnostics and Maintenance;
- Information from other vehicles.

Moreover, it considers 3 categories of non-safety-related application scenarios:

- Traffic Management;
- Tolling;
- Information from other vehicles.

A preliminary report [4] classified the scenarios included in the previous categories with regard to five classes of *allowable latencies*: 20 milliseconds, 100 milliseconds, 500 milliseconds, 1 second and 5 seconds. Furthermore, a following report [24] also identified eight high-priority scenarios that are critical for safety guarantees:

- Traffic Signal Violation Warning: $100ms$ of latency with a one-way infrastructure-to-vehicle communication;
- Curve Speed Warning: $1s$ of latency with a one-way infrastructure-to-vehicle communication;
- Emergency Electronic Brake Light: $100ms$ of latency with a one-way vehicle-to-vehicle communication;

TABLE IV
TIMINGS FOR THE DIFFERENT SCENARIOS WITH CURVE *secp256r1*[ms]

	FS	DS	R
x86_64	1.25	0.32	0.69
RPi4	5.61	1.44	3.02
RPi3	11.46	3.05	6.39
RPi1	58.88	15.38	31.94
M4	459.20	118.00	253.60
M3	622.00	161.00	344.00

TABLE V
TIMINGS FOR THE DIFFERENT SCENARIOS WITH CURVE *secp256k1*[ms]

	FS	DS	R
x86_64	0.93	0.25	0.48
RPi4	4.60	1.21	2.32
RPi3	9.55	2.53	4.80
RPi1	51.04	13.15	27.17
M4	362.55	93.31	195.64
M3	564.00	144.00	304.00

- Pre-Crash Sensing: $20ms$ of latency with a two-way vehicle-to-vehicle communication;
- Cooperative Forward Collision Warning: $100ms$ of latency with a one-way vehicle-to-vehicle communication;
- Left Turn Assistant: $100ms$ of latency with a one-way vehicle-to-infrastructure and infrastructure-to-vehicle communication;
- Lane changing Warning: $100ms$ of latency with a one-way vehicle-to-vehicle communication;
- Stop Sign Movement Assistance: $100ms$ of latency with a one-way vehicle-to-infrastructure and infrastructure-to-vehicle communication;

To investigate when the proposed implementation is suitable to the considered scenarios we estimate the timings required to compute all the due cryptographic operations for *sending* and for *receiving* a message, including both the ECQV and the ECDSA schemes. In particular, we consider that sending a message requires the generation of a novel ECQV certificate and the computation of an ECDSA signature, while the reception of a message requires the extraction of the public key from the ECQV certificate and the verification of the ECDSA signature.

To understand the feasibility of the different devices in realistic scenarios, we also take into account that each device might be used in specific operational model, where they only operate a certain set of operations. Thus, we define two device roles: **sender (S)** and **receiver (R)**. The sender role is further distinguished in two sub-roles: the **full sender (FS)**, which identifies a device that generates a novel certificate for each communication, as used in the Butterfly protocol [25], and the **direct sender (DS)**, which identifies a device that requests a valid certificate offline and uses the same certificate for multiple communications. The operations required in the **FS** role are the *CsrGen*, *Validate*, and *Sign*, while the only required operation in the **DS** role is the *Sign* operation. Receiver devices must extract the public key from the certificate and verify the validity of the signature, thus requiring the execution of the *Extract* and the *Verify* operations. Tables IV and V show the timings required for the cryptographic operations with regard to each role and compare them against the five NHTSA *allowable latencies* profiles by using the *curve256r1* and *curve256k1* curves, respectively.

Tables VI and VII show the summary of the constraints of the boards with regard to the different role by using the *secp256r1* and *secp256k1* curves, respectively. The rows of Tables VI and VII represent the allowable delays accepted by the standard, the columns represent the different platforms, and each sub-column is used to highlight the boards limitations. The FS and DS sub-columns are used to present the applicability of the board as a sender device with regard to send rates, that are fixed and defined by the standards. The R sub-column shows the maximum number of messages that can be validated by the platform in a receiving scenario within the allowable delay. We remark that this is the maximum number of messages, and that the actual number of received messages depends from the number of vehicles within communication range. With the latency requirement of 20 milliseconds it is possible to deploy the x86_64, the RPi4, and the RPi3 architectures with the FS role, while in the DS role it is possible to also deploy the RPi1 board. By using the *secp256r1* curve in the R role, only the x86_64, RPi4, and RPi3 systems allow to verify the signatures of 29, 6, and 3 incoming messages within the maximum allowed latency, while the same boards allow to verify 41, 8 and 4 signatures by using the *secp256k1* curve. In safety-critical applications with a latency requirement of 100 milliseconds, it is possible to use the x86_64, RPi4, RPi3, and RPi1 systems for all roles and curves, while the M4 can only be used as DS by using the *secp256k1* curve. The maximum number of messages that the boards are able to verify within the required $100ms$ latency requirement in the R role are 144, 33, 15, and 3 with the *secp256r1* curve, while the same boards can verify 209, 43, 20 and 3 signatures with using the *curve256k1* curve. With a $500ms$ latency requirement, in the FS role it is possible to deploy the same boards deployed in case of $100ms$ plus the M4, while in both the DS and R roles it is possible to use all boards for both curves. The maximum number of messages that each board is able to verify within the required time are 724 (x86_64), 165 (RPi4), 78 (RPi3), 15 (RPi1), 2 (M4) and 1 (M3) with the *curve256r1* and 1048 (x86_64), 215 (RPi4), 104 (RPi3), 18 (RPi1), 2 (M4) and 1 (M3) with the *curve256k1*. With higher required latency it is possible to use any board in any role it is possible to use board like the M3 and the M4 in scenarios with low network traffic.

V. CONCLUSIONS

This paper proposes an experimental analysis of the ECQV performance in VANETs communications. Its two main contributions to the state-of-the-art are: (i) the proposal of the first open reference implementation of ECQV tailored for embedded devices and automotive-grade boards [3] and the

TABLE VI
ANALYSIS OF THE CONSTRAINTS OF THE BOARDS. APPLICABILITY FOR BOTH SENDER ROLES AND MAXIMUM NUMBER OF RECEIVED MESSAGES FOR THE RECEIVER ROLE (SECP256R1)

	x86_64			**RPi4**			**RPi3**			**RPi1**			**M4**			**M3**		
	FS	DS	R	FS	DS	R	FS	DS	R	FS	DS	R	FS	DS	R	FS	DS	R
20 ms	✓	✓	29	✓	✓	6	✓	✓	3	×	✓	×	×	×	×	×	×	×
100 ms	✓	✓	144	✓	✓	33	✓	✓	15	✓	✓	3	×	×	×	×	×	×
500 ms	✓	✓	724	✓	✓	165	✓	✓	78	✓	✓	15	✓	✓	2	×	✓	1
1000 ms	✓	✓	1449	✓	✓	331	✓	✓	156	✓	✓	31	✓	✓	3	✓	✓	2
5000 ms	✓	✓	7246	✓	✓	1657	✓	✓	783	✓	✓	156	✓	✓	19	✓	✓	14

TABLE VII
ANALYSIS OF THE CONSTRAINTS OF THE BOARDS. APPLICABILITY FOR BOTH SENDER ROLES AND MAXIMUM NUMBER OF RECEIVED MESSAGES FOR THE RECEIVER ROLE (SECP256K1)

	x86_64			**RPi4**			**RPi3**			**RPi1**			**M4**			**M3**		
	FS	DS	R	FS	DS	R	FS	DS	R	FS	DS	R	FS	DS	R	FS	DS	R
20 ms	✓	✓	41	✓	✓	8	✓	✓	4	×	✓	×	×	×	×	×	×	×
100 ms	✓	✓	209	✓	✓	43	✓	✓	20	✓	✓	3	×	✓	×	×	×	×
500 ms	✓	✓	1048	✓	✓	215	✓	✓	104	✓	✓	18	✓	✓	2	×	✓	1
1000 ms	✓	✓	2096	✓	✓	431	✓	✓	208	✓	✓	36	✓	✓	5	✓	✓	3
5000 ms	✓	✓	10482	✓	✓	2155	✓	✓	1041	✓	✓	184	✓	✓	25	✓	✓	16

evaluation of the times required to carry out the main cryptographic operations on five different embedded platforms; (ii) analytic evaluations that identify the boards that can be used to implement different roles in secure V2V communications. Results highlight that the timing requirements mandated by the NHTSA report require a careful evaluation of the hardware devices that will have to support secure communications in current and future C-ITS.

Acknowledgment The research leading to these results has received funding from the ECSEL JU programme under the SECREDAS Project grant agreement nr. 783119.

REFERENCES

[1] C. K. Toh, "Ad hoc wireless networks: Protocols and systems," USA, Tech. Rep., 2001.

[2] A. Paul, N. Chilamkurti, A. Daniel, and S. Rho, "Chapter 2 - intelligent transportation systems," in *Intelligent Vehicular Networks and Communications*. Elsevier, 2017.

[3] F. Pollicino, D. Stabili, L. Ferretti, and M. Marchetti. Implementation of the ECVQ implicit certificate scheme for low-power devices. [Online]. Available: https://weblab.ing.unimore.it/resources/uECQV.zip

[4] National Highway Traffic Safety Administration, "Vehicle safety communication projet – task 3 final report," DOT HS 809 859, March 2005.

[5] N. Nowdehi, W. Aoudi, M. Almgren, and T. Olovsson, "CASAD: CAN-Aware Stealthy-Attack Detection for In-Vehicle Networks," arXiv 1909.08407, 2019.

[6] G. Dupont, J. den Hartog, S. Etalle, and A. Lekidis, "A survey of network intrusion detection systems for controller area network," in *Proc. IEEE Int'l Conf. Vehicular Electronics and Safety*, Sep. 2019.

[7] D. Stabili and M. Marchetti, "Detection of missing can messages through inter-arrival time analysis," in *Proc. IEEE 90th Vehicular Technology Conf.*, Sep. 2019.

[8] M. Marchetti and D. Stabili, "READ: Reverse Engineering of Automotive Data Frames," *IEEE Trans. Information Forensics and Security*, vol. 14, no. 4, 2019.

[9] M. D. Pesé, T. Stacer, C. A. Campos, E. Newberry, D. Chen, and K. G. Shin, "Librecan: Automated can message translator," in *Proc. 2019 ACM SIGSAC Conf. Computer and Communications Security*, 2019.

[10] D. Stabili, L. Ferretti, and M. Marchetti, "Analyses of secure automotive communication protocols and their impact on vehicles life-cycle," in *Proc. IEEE Int'l Conf. Smart Computing*, June 2018.

[11] T. Rosenstatter, C. Sandberg, and T. Olovsson, "Extending AUTOSAR's Counter-Based Solution for Freshness of Authenticated Messages in Vehicles," in *Proc. IEEE 24th Pacific Rim Int'l Symp. Dependable Computing*, Dec 2019.

[12] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, and J.-P. Hubaux, "Secure vehicular communication systems: design and architecture," *IEEE Comm. Magazine*, vol. 46, no. 11, 2008.

[13] F. Kargl, P. Papadimitratos, L. Buttyan, M. Müter, E. Schoch, B. Wiedersheim, T.-V. Thong, G. Calandriello, A. Held, A. Kung *et al.*, "Secure vehicular communication systems: implementation, performance, and research challenges," *IEEE Comm. Magazine*, vol. 46, no. 11, 2008.

[14] M. Raya and J.-P. Hubaux, "The security of vehicular ad-hoc networks," in *Proc. 3rd ACM Work. Security of ad-hoc and sensor networks*, Jul. 2005.

[15] M. A. R. Baee, L. Simpson, E. Foo, and J. Pieprzyk, "Broadcast authentication in latency-critical applications: On the efficiency of ieee 1609.2," *IEEE Trans. Vehicular Technology*, vol. 68, no. 12, 2019.

[16] IEEE, "Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages," Std. 1609.2a-2017.

[17] ETSI, "ITS-G5 access layer specification for intelligent transport systems operating in the 5 ghz frequency band," EN 302 663, 2019.

[18] Association of Radio Industries and Business, "700 mhz band intelligent transport systems," ARIB STD-T109, 2012.

[19] C. Research, "Sec 4: Elliptic curve qu-vanstone implicit certificate scheme, standards for efficient cryptography group. version 1.0," 2013.

[20] K. MacKay, "MicroECC," https://github.com/kmackay/micro-ecc, Last visited Mar. 2020.

[21] Federal Information Processing Standard, "Digital Signature Standard (DSS)," National Institute of Science and Technology, FIPS 186-4, 2013.

[22] E. Brier and M. Joye, "Weierstraß elliptic curves and side-channel attacks," in *Proc. Int'l Work. public key cryptography*, Feb. 2002.

[23] C. Research, "Standards for EfficientCryptography 1," SEC 1, 2009.

[24] National Highway Traffic Safety Administration, "Vehicle safety communication projet – final report," DOT HS 810 591, Apr. 2006.

[25] M. A. Simplicio, E. L. Cominetti, H. K. Patil, J. E. Ricardini, and M. V. M. Silva, "The unified butterfly effect: Efficient security credential management system for vehicular communications," in *Proc. 2018 IEEE Vehicular Networking Conf.*, Dec 2018.