Accepted copy for Publication at the Design, Automation and Test in Europe (DATE) Conference 2023 Final published version available at: https://doi.org/10.23919/DATE56975.2023.10137025

# Establishing Dynamic Secure Sessions for ECQV Implicit Certificates in Embedded Systems

Fikret Basic, Christian Steger Institute of Technical Informatics Graz University of Technology Graz, Austria {basic, steger}@tugraz.at Robert Kofler R&D Battery Management Systems NXP Semiconductors Austria GmbH Co & KG Gratkorn, Austria robert.kofler@nxp.com

Abstract—Be it in the IoT or automotive domain, implicit certificates are gaining ever more prominence in constrained embedded devices. They present a resource-efficient security solution against common threat concerns. The computational requirements are not the main issue anymore. The focus is now placed on determining a good balance between the provided security level and the derived threat model. A security aspect that often gets overlooked is the establishment of secure communication sessions, as most design solutions are based only on the use of static key derivation, and therefore, lack the perfect forward secrecy. This leaves the transmitted data open for potential future exposures by having keys tied to the certificates rather than the communication sessions. We aim to patch this gap, by presenting a design that utilizes the Station to Station (STS) protocol with implicit certificates. In addition, we propose potential protocol optimization implementation steps and run a comprehensive study on the performance and security level between the proposed design and the state-of-the-art key derivation protocols. In our comparative study, we show that with a slight computational increase of 20% compared to a static ECDSA key derivation, we are able to mitigate many session-related security vulnerabilities that would otherwise remain open.

*Index Terms*—ecqv, implicit, certificate, sts, dynamic, session, key derivation, embedded, security, constrained, automotive.

#### I. INTRODUCTION

Security is becoming increasingly important in protecting the ever-expanding connections of modern embedded devices. The use of common schemes, e.g., Transport Layer Security (TLS), often proves to be difficult due to the constrained nature of the used devices, which can only allow for a limited performance overhead [1]. In contrast, implicit certificates are showing promise in replacing the traditional security architecture schemes. Implicit certificates offer a lightweight certificate format, and a flexible public key derivation and authentication mechanism that make the use of public key infrastructures more accessible for constrained embedded systems [2]–[6].

Different schemes exist based on the implicit certificates, with Elliptic Curve Qu-Vanstone (ECQV) still being the most popular and researched one [7]. While there has been numerous research done on ECQV and its use with embedded systems [2]–[6], [8]–[10], we noticed that certain security aspects are left out when considering the session key derivation process. The key derivation (KD) and session establishment solutions often neglect a very important key aspect, the *perfect forward secrecy*, specifically, the ephemeral key security characteristic. Forward secrecy allows for a dynamic KD and it considers the state where each newly derived key has a high-enough entropy and is independent of a previous one [11]. This is especially important in session communication, where interactions happen on a frequent basis. We believe that is characteristic often gets neglected due to a believed premise of the necessity for sacrificing the security strength for the performance gain with the limited embedded devices. Rather, what often gets deployed is a static KD where key computations are directly linked to their certificate material. These keys would, hence, only be changed by the change of the certificates and through re-initiating the authentication and session establishment steps. It is, therefore, called a static key exchange, since no other KD function or additional input data is used to mask the present session key which is fully dependent on the current certificate. This can be very problematic in situations where, implementation-wise, either due to the limitations in the system's architecture, constrained nature of the devices, or neglect from the developers, can lead to longer than the intended use of the same session key.

Regular key updates are important, as in unfortunate cases, where the session key might get compromised, e.g., via the node capturing attack by compromising a valid device that holds it, all the captured exchanged messages would also be able to be decrypted. Any attack that can compromise the stored device credentials would be able to exploit the statically derived keys. An especially dangerous attack, which is also prevalent in TLS, is the key compromise impersonation (KCI). It is a man-in-the-middle (MitM) attack where an attacker can impersonate the trusted server side to manipulate the key derivation process [12]. In 2018, OWASP rated for internet of things (IoT) weak, guessable and hard-coded passwords as the number one weakness for the IoT systems, which also considers the key credentials [13]. In fact, based on the study by the SEC Consult between 2015 and 2016, the number of exposed private keys by IoT devices grew by 40% [14]. The ENISA initiative, targeted at investigating automotive security vulnerabilities, listed remote attacks, theft and surveillance as one of the most potent attacks that can happen due to the lack of the required cryptographic functionality support. In their document, all three attacks are affiliated with the lack of forward secrecy for both the wide and local networks [15].



Fig. 1. Centralized implicit certificate architecture.

To mitigate these security vulnerabilities, we focus on providing a solution that is independent of the rate of the certificate updates, and which ensures that each new communication session would always yield a new key derivation. Additionally, we want to make sure that a session key compromise does not lead to exposure of previous or further keys, i.e., to guarantee perfect forward secrecy. To fulfil these constraints, we present a design based on the Station-to-Station (STS) protocol [11] for a dynamic KD for implicit certificate schemes and extend on the general lightweight ECQV implementation by Pollicino et al. [2]. Furthermore, we investigate the optimization steps for the STS KD protocol execution for the implicit certificates, analyze its applicability for the embedded hardware by implementing and evaluating it on different devices, and compare it with other related implicit certificate schemes. Summarized, our main contributions contained within this work are:

- Design and implementation of a dynamic key derivation approach for implicit certificate architecture schemes using the STS protocol.
- Performance and security evaluation of state-of-the-art (SotA) KD implicit certificate schemes by expanding on the existing work from the automotive and IoT domains.
- 3) Testing the protocol's feasibility in an automotive system by implementing it on top of a battery management system (BMS) to depict a real-world scenario.

#### **II. BACKGROUND ON THE SECURITY ARCHITECTURE**

We consider three main stages when deploying implicit certificates in a network, as shown in Figure 1 [5], [8]: (1) device authentication and deployment, (2) certificate derivation, and (3) session establishment. The deployment phase primarily depends on the main system architecture, however, it generally contains a central, and a more powerful, certificate authority (CA) device. The certificate derivation phase is straightforward with ECQV and almost identical among different solutions [3], [5], [6], [8]. The session establishment process often differs and depends on the KD and node authentication algorithms.

## A. Key derivation for secure sessions

We differentiate between two sessions, the certificate session and the communication session. The certificate session considers the validity duration of the currently issued certificates, e.g., in a vehicle during each new engine start, while the communication session considers the duration during one message exchange between two or multiple devices, e.g., monitoring, updates, status readout, etc.

We refer to static key derivation (SKD) as the calculation approach that relies on the traditional Diffie-Hellman KD, i.e., where the keys or the underlying secret are derived from the multiplication of the stored private key and the other device's public key as  $S_k = Prk_a * Puk_b = Prk_b * Puk_a$ . The SKD secret is tied to its current certificate session rather than the communication session. As long as the private and public key pairs are not updated, the underlying session key will also not change. Contrarily, the dynamic key derivation (DKD), as the one presented in this work, fulfils the condition that a new session key is derived on each new communication session start, regardless of the current certificate session. The DKD makes sure that each communication session remains independent from the other sessions and should, ideally, provide the perfect forward secrecy attribute. A key derived via this method is also known as the ephemeral secret key.

#### III. RELATED WORK

Several research works have already been published on the use of the ECQV and the session KD, both under the general and embedded environments. Porambage et al. [3], [9] present one of the earlier session authentication and key exchange solutions for the wireless networks, where the communication between the nodes is done using an SKD. For authentication, the protocol uses Message Authentication Code (MAC) with pre-embedded keys, but it also requires that each node possesses from each other the authentication key. A different authentication scheme is presented by Siddhartha et al. [6], where an "authenticator" is used. It is made out of certificaterelated data and signed by the CA. A hash function is also used for the additional integrity check. The session key calculation, however, is still based on the standard SKD.

D. Lee and I. Lee [16] present two approaches to KD in a constrained IoT environment. The first approach is based on the pure ECQV methodology with no additional authentication steps. It relies on validating the identification (ID) and correctness of the certificate calculation, but this does not guarantee the authenticity of the device itself. The certificates and the ID could be spoofed, resulting in a false identification by a malicious actor. Additionally, similar to the work presented by Sciancalepore et al. [4], the KD uses additional nonces to diversify the key. However, this does not add additional protection since the underlying secret is still calculated using an SKD, i.e., it only considers the multiplication of the private and public keys. The nonces used in the KD can be read from a monitored network. Their second method does provide DKD and ephemeral keys, nonetheless, both methods suffer from a central problem, and that is that the device authentication is not considered, rather only the public key validity.

Recently, Zi-Yuan Liu et al. [10] presented an extension of the ECQV, where devices might house multiple certificates and keys. While novel, the challenges presented in the paper are currently not relevant for this work's use cases, as the focus is placed on larger dynamic networks.



Fig. 2. Key derivation using STS protocol for ECQV architectures.

#### IV. A NOVEL DYNAMIC KEY DERIVATION FOR ECQV

#### A. Security requirements

For the security requirements, it is intended to provide a design that can answer to the following threats: (T1) past data exposure, (T2) MitM attacks, (T3) node capturing attacks, (T4) key data reuse for further session calculations, (T5) key derivation exploitation; each unique key needs to have a high-enough entropy, and that is only stored, and being able to be stored, by the valid parties. We aim to protect two important system assets: session data, and security credentials. The design also needs to be lightweight in its implementation so as to be easily accessible for the embedded devices.

#### B. Protocol formalization

We base our design of the DKD on the use of the STS protocol [11], [17]. STS is a known protocol used in wide networks; however, it has not been previously investigated for use with the ECQV. The STS derivation should consider the ECQV implicit certificate calculation properties. The protocol steps are shown in Figure 2. It is assumed that the first two phases are correctly done as explained in Section II.

The protocol uses the implicit certificate with the elliptic curve digital signature algorithm (ECDSA) to provide authentication as shown with Algorithm 1, and verification with Algorithm 2. What makes it unique compared to other STS algorithm derivations, is that ECQV relies on the implicit derivation of the public key for the signature verification. The security of the ECDSA algorithm with the ECQV scheme has been proven secure against passive attacks [18]. The public key calculation used for verification is derived as:

$$Q_X = Hash(Cert_X) * Decode(Cert_X) + Q_{CA} \quad (1)$$

The STS provides ephemeral keys, by always deriving a new random elliptic curve (EC) point in the request as:

$$X \in_R [1, \dots, n-1] \to XG = X * G \tag{2}$$

Derivation of session  $K_S$  keys is done by calculating:

$$K_{PM} = X_A * XG_B = X_B * XG_A \tag{3}$$

$$K_S = KDF(K_{PM}, salt) \tag{4}$$



Fig. 3. Time duration of individual STS operation runs on an STM32F676.

Algorithm	1:	STS	implicit	certificate	auth.	response.	
			1			1	

Input:  $XG_A, XG_B, K_S$ Output: Resp if  $device_A$  then  $d_{sign} \leftarrow sign(Prk_A, (XG_A || XG_B))$ 2 else 3  $\leftarrow sign(Prk_B, (XG_B||XG_A))$ 4  $d_{sign}$ end 5  $Resp \leftarrow$  $encrypt(K_S, d_{sign})$ 6 7 return Resp

Algorithm 2: STS implicit certificate sign. verification.

Input:  $Resp_X, Cert_X$ Output:  $Status_{Ok}, Status_{Err}$ 1  $d_{sign_X} \leftarrow decrypt(K_S, Resp_X)$ 2  $Q_X \leftarrow hash(Cert_X) * decode(Cert_X) + Q_{CA}$ 3  $Status \leftarrow verify(Q_X, d_{sign_X})$ 

4 return Status

#### C. STS protocol optimization

Even though the STS protocol might provide more security advantages compared to related KD implicit certificate protocols (see Section V-D), the main drawback is in its timely execution. As this is still an important aspect of modern constrained systems, we investigate potential optimizations. We divide the entire STS ECQV protocol into four operations:

- Op1 Request phase; random XG point derivation
- Op2 Public key and premaster session key generations
- Op3 Auth. signature derivation and encryption
- Op4 Auth. signature decryption and verification

In this analysis, we do not consider the transfer time. We derive two potential optimizations. Similar to the work presented by Sciancalepore et al. [4], the initial request can be made to contain both the certificate and the XG data, with the calculations of the public key and premaster secret data (see Op2) being done in parallel. Further optimization could be to also include the following Op3 to be executed parallel after Op2 as well. There is a drawback here, and that comes at the expense of the algorithm's flexibility. Failed authentication requests would only be checked after the calculations have been processed. This could open some doors for misuse by malicious users, either through denial-of-service, or similar attacks. But the actual implementation does not suffer in terms of general security since the calculations are still processed in the same manner. The main advantage would be from the system design perspective, which would allow additional operations to run in parallel. The sent data is identical to the original protocol, but the message and content order vary slightly. Figure 3. shows individual operation time requirements.



Fig. 4. Comparison of the total KD protocols processing time.

The total execution time with the conventional STS between two devices can be represented as:

$$\tau_T = \sum_{i=1}^{N_{Op}} T_{OpAi} + \sum_{i=1}^{N_{Op}} T_{OpBi} \text{ ,with } N_{Op} = 4 \qquad (5)$$

As the optimization can be applied through the Op2 and Op3, we get the following derivation based on the time that each device takes to calculate the operations:

$$\forall x \in \{2,3\}, T_{OpAx} = \begin{cases} 0, & \text{if } A = B\\ |T_{OpAx} - T_{OpBx}|, & \text{otherwise} \end{cases}$$
(6)

This means that no additional time is taken per device A (or B, as it is symmetrical) if they are identical, or if they are not, the extra amount of time depends on the difference in their execution time for Op2 and Op3.

If the devices are equal, ideally, optimization formulas for two different steps of optimizations based on the system requirements would bring the total run times to:

Opt. I 
$$\tau_T = 2 * T_{Op1} + T_{Op2} + 2 * T_{Op3} + 2 * T_{Op4}$$
 (7)

Opt. II 
$$\tau_T'' = 2 * T_{Op1} + T_{Op2} + T_{Op3} + 2 * T_{Op4}$$
 (8)

The primary advantage of the optimization is the clear reduction in the total execution time by maintaining a minimal change to the original STS protocol structure. In Section V-A, we compare different protocols for the implicit certificate KD and show the difference in time execution between the optimized and non-optimized STS on real embedded hardware.

## V. IMPLEMENTATION AND EVALUATION

## A. Protocol performance evaluation

.,

To show the feasibility of the proposed STS protocol derivation in modern systems and compare it with other SotA KD protocols for implicit certificates, we implement and run the protocols under different embedded devices. We analyze the runs under three main hardware performance level groups:

- Low-end: Arduino, ATmega2560, 8-bit 16MHz
- Mid-tier: S32K144, ARM Cortex-M4F 32-bit 80MHz; and STM32F767, Cortex-M7 32-bit 216MHz
- High-end: Raspberry Pi 4, Cortex-A72 64-bit 1.5GHz

The implementations are done in C and make use of the functions provided by the micro-ecc, tiny-aes, and bear-ssl libraries, as well as the micro ECQV functions provided by Pollicino et al. [2]. All protocols have been tested with the secp256r1 256-bit EC, with 256-bit level for the SHA and HMAC, and 128-bits for the AES and CMAC.



Fig. 6. CAN-FD network layers used for the session test communication.

In total, we test four different protocols derived from two groups based on the use of the authentication mechanism, i.e., on those that rely on the use of ECDSA: (i) static ECDSA by Basic et al. [5] as S-ECDSA, and (ii) STS from this work, and those that only use the symmetric cryptography authentication without the EC operations: (iii) from Porambage et al. [3] as PORAMB, and (iv) from Sciancalepore et al. [4] as SCIANC. We also consider the extension of the S-ECDSA protocol, specifically the additional authentication of the ack acknowledgement messages, based on the finished message handling as seen from Porambage et al. [3]. Furthermore, we also evaluate the STS protocol when considering the optimization steps explained in Section IV-C. Only STS is the true DKD, while the rest fall into the SKD category. The results of the evaluation are shown in Table I, with Figure 4. showing the graphical representation for the STM32F767. The times are averaged after ten runs. The measurements were done using system ticks and Nordic PPK2. The run time scalability is relatively consistent regarding the devices' performances. While STS shows the highest execution time, its optimization variants show the potential time similar to or faster than the S-ECDSA. The PORAMB and SCIANC show the fastest time as they use a different authentication mechanism and do not rely on the EC operations. However, these protocols lack some of the necessary security options as discussed in Section V-D.

#### B. Overhead examination

To give a clearer analysis of the algorithm processing time, it would be advantageous to consider the transmission overhead, however, that parameter is heavily dependent on the used communication protocol and its configuration. Here, we provide an overview of the overhead for each algorithm during the KD exchange protocol, independent of the communication technology in use. We consider only the protocol-affiliated transmission data on the application level. Security algorithms bit sizes are the same as the ones used in Sect. V-A. We assume IDs to be of 16 bytes and use the minimal certificate encoding with 101 total bytes [7]. The results are shown in Table II.

Both the S-ECDSA and STS protocols showed similar transmission sizes, with also the least communication steps when 
 TABLE I

 Execution time in milliseconds of the KD protocols for ECQV for the respective embedded hardware.

Protocol / Device	ATMega2560	S32K144	STM32F767	RaspberryPi 4
S-ECDSA	$36859.26 \pm 0.18$	$2894.1\pm9.83$	$2521.77 \pm 5.87$	$18.76\pm0.11$
S-ECDSA (ext.)	$36882.64 \pm 0.23$	$2976.2 \pm 11.56$	$2602.69 \pm 8.61$	$18.68\pm0.12$
STS	$46262.03 \pm 0.13$	$3622.71 \pm 7.034$	$3162.07 \pm 7.52$	$23.26\pm0.12$
STS (opt. I)	$41680.23 \pm 1.2$	$3246.55 \pm 12.97$	$2818.02 \pm 11.26$	$20.87 \pm 0.07$
STS (opt. II)	$32410.81 \pm 1.14$	$2556.84 \pm 13.13$	$2219.25 \pm 11.3$	$16.31\pm0.07$
SCIANC	$8990.49 \pm 0.03$	$721.67\pm0.28$	$628.1\pm0.32$	$4.58\pm0.02$
PORAMB	$17932.17 \pm 0.05$	$1471.66 \pm 0.63$	$1263.0\pm0.42$	$8.98\pm0.04$

 TABLE II

 COMMUNICATION STEPS AND TRANSMISSION OVERHEAD OF THE KD PROTOCOLS FOR ECQV.

Protocol	S-ECDSA(+ext.)	STS	SCIANC	PORAMB
Step: Op. (X bytes)	A1: ID(16), Nonce(32) B1: ID(16), Cert(101),	A1: ID(16), XG(64) B1: ID(16), Cert(101),	A1: ID(16), Nonce(32), Cert(101) B1: ID(16),	A1: Hello(32), ID(16)
	Sign(64), Nonce(32) A2: Cert(101), Sign(64)	XG(64), Resp(64) A2: Cert(101), Resp(64)	Nonce(32), Cert(101) A2: Auth_MAC(32)	A2: Cert(101), Nonce(32), MAC(32)
	B2: ACK(1), (+Ext_Fin(96)) A3: (+Ext_Fin(96))	B2: ACK(1)	B2: Auth_MAC(32)	B2: Cert(101), Nonce(32), MAC(32) A3 & B3: Finish(197)
Total	4(+1): 427(+192) B	4: 491 B	4: 362 B	6: 820 B

not considering the last ack message. The SCIANC protocol also requires only four transmissions, but with only 362 total bytes under the assumed setup. Contrarily, the PORAMB algorithm showed the largest overhead, with 6 total steps and 820 bytes. We did not include the optimized version of STS since it does not differ in terms of the transmitted data. Considering the fast data rates of most communication protocols and the presented data sizes, we can conclude that the influence of the transmission overhead would be minimal in comparison to the individual KD protocols. This is further complemented by the prototype evaluation results from Section V-C.

## C. Prototype implementation evaluation

In order to evaluate the proposed protocol design on its technical use, we implemented a prototype system that depicts a common communication occurrence between two ECUs in an automotive network. It handles the secure communication between a BMS controller, and an electric vehicle charging controller (EVCC) [19]. Both devices are represented with an S32K144 microcontroller from the NXP Semiconductors to portray a real-world environment. The BMS is additionally connected to a battery cell controller and a battery emulator for emulating a functional unit. The setup is shown in Figure 5.

The session communication between the devices takes place over a Controller Area Network (CAN) interface. The test suite uses the CAN-FD derivation with an implemented CAN-TP layer for message fragmentation [20]. Figure 6 shows the message formats. The devices also communicate with a more



Fig. 7. Timeline model of the prototype session communication between a BMS and EVCC for: (A) STS & (B) S-ECDSA, ECQV KD protocols.

powerful CA gateway (represented with a Raspberry Pi 4) to handle the initial device authentication and certificate distribution. The nominal phase CAN-FD bit rate was configured at 0.5 Mbit/s, with the data phase rate being set at 2 Mbit/s.

For the evaluation, we compare the proposed STS implementation against the common static ECDSA [2], [5]. For a fair comparison, as to account for the conventional deployment of these protocols in the field, we did not consider the optimization handling for the parallel operation runs argued in Section IV-C. The implemented security protocols use the same library sources as those mentioned in Section V-A. The timeline of both protocols is shown in Figure 7. The

 TABLE III

 Security overview of the KD protocols for ECQV.

	S-ECDSA	STS	SCIAN	C PORAMB	
Data exposure	X	$\checkmark$	X	X	
Node capturing	$\Delta$	$\Delta$	X	X	
Key data reuse	X	$\checkmark$	$\Delta$	X	
Key der. exploit	$\Delta$	$\checkmark$	$\Delta$	$\Delta$	
Auth. procedure	$\checkmark$	$\checkmark$	$\Delta$	$\Delta$	
Session Security Data Credentials					
Exposure	Attacks Cap	Node	Reuse	Exploitation	
		$\overline{}$			
[C1] Forward Secrecy [C2] ECDSA Authentication Protection [C3] STS & ECQV Property					

Fig. 8. Block diagram representation for the STS-ECQV KD threat model.

STS implementation showed only a slight difference in the total run time with 3.257 s compared to S-ECDSA's 2.677 s, i.e., an increase of 21.67 %. The CAN-FD transfer time over the physical link was negligible (< 1 ms). The majority of the communication time from Figure 7. was for the data processing on the remaining layers.

## D. Security analysis

We concern ourselves with the listed threats from Section IV-A and compare the previous KD algorithms on the provided security level. We also specially look at the mutual authentication procedure, as an important feature against MitM attacks. The analysis is presented in Table III, with the following notation: X - weak or no countermeasure,  $\Delta$  - partial protection,  $\checkmark$  - fully protected.

The lack of forward secrecy for all protocols, except STS, makes them highly vulnerable to previous session data exposure, key material reuse (while having the same certificates), and node-capture attacks. However, we note that no algorithm is fully protected against the node-capture attacks, as even with STS, the protection can only be guaranteed for the previous messages, not the future ones. The mutual authentication for both SCIANC and PORAMB is based on symmetric cryptography with some concerns. PORAMB has the requirement to store individual keys per the number of devices, which makes future updates troublesome. SCIANC algorithm ties its session key with the KD authentication, meaning that if the session key gets exploited so will the future authentication. On the other hand, with S-ECDSA and STS, the authentication is based on the ECDSA with private keys used for signature derivation. Figure 8. shows the derived countermeasures on the listed threats for the STS-ECQV KD.

## VI. CONCLUSION

In this work, we have presented a key derivation and session establishment model using the STS protocol within the ECQV implicit certificate framework, and its relation and comparison with other KD protocols on embedded devices. While requiring more time, the STS offers a good balance between providing additional security features and certainty without compromising much of the performance. It showed a slight run time increase of  $\approx 21\%$  compared to a static ECDSA KD protocol, with no additional communication overhead. While other non-EC authentication-based KD protocols showed a noticeable faster execution time, they also lacked the security level acceptable for modern systems. To compensate for the STS run time, we introduced a series of optimization steps for the protocol operations. For future work, we plan to investigate the influence of security modules and hardware accelerators when considering the implicit certificate protocols on embedded devices, especially those related to session establishment.

#### ACKNOWLEDGMENT

This project has received funding from the "EFREtop: Securely Applied Machine Learning - Battery Management Systems" (Acronym "SEAMAL BMS", FFG Nr. 880564).

#### REFERENCES

- J. P. Hughes and W. Diffie, "The Challenges of IoT, TLS, and Random Number Generators in the Real World: Bad Random Numbers Are Still with Us and Are Proliferating in Modern Systems.," *Queue*, jun 2022.
- [2] F. Pollicino et al., "An experimental analysis of ECQV implicit certificates performance in VANETs," in *IEEE 92nd VTC2020-Fall*, 2020.
- [3] P. Porambage et al., "Two-phase authentication protocol for wireless sensor networks in distributed iot applications," in IEEE WCNC, 2014.
- [4] S. Sciancalepore, G. Piro, G. Boggia, and G. Bianchi, "Public Key Authentication and Key Agreement in IoT Devices With Minimal Airtime Consumption," *IEEE Embedded Systems Letters*, vol. 9, 2017.
- [5] F. Basic et al., "Trust your BMS: Designing a Lightweight Authentication Architecture for Industrial Networks," in 23rd IEEE ICIT, 2022.
- [6] V. Siddhartha, G. S. Gaba, and L. Kansal, "A Lightweight Authentication Protocol using Implicit Certificates for Securing IoT Systems," *Procedia Computer Science*, vol. 167, pp. 85–96, 2020.
- [7] M. Campagna, Standards for Efficient Cryptography 4 (SEC4): Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV). Certicom Corp., 2013.
- [8] D. Puellen et al., "Using Implicit Certification to Efficiently Establish Authenticated Group Keys for In-Vehicle Networks," in Proc. of the 11th IEEE VNC Conf., 2019.
- [9] P. Porambage *et al.*, "Certificate-Based Pairwise Key Establishment Protocol for Wireless Sensor Networks," in *16th IEEE CSE*, 2013.
- [10] Z.-Y. Liu et al., "Extension of Elliptic Curve Qu–Vanstone Certificates and Their Applications," J. Inf. Secur. Appl., vol. 67, jun 2022.
- [11] W. Diffie et al., "Authentication and Authenticated Key Exchanges," Design, Codes and Cryptography, p. 107–125, June 1992.
- [12] C. Hlauschek et al., "Prying Open Pandora's Box: KCI Attacks against TLS," in 9th USENIX WOOT, (Washington, D.C.), Aug. 2015.
- [13] OWASP, "OWASP IoT Top 10." https://owasp.org, 2018.
- [14] S. Consult, "House Of Keys: 9 Months Later... 40% Worse." https://secconsult.com/blog/detail/house-of-keys-9-months-later-40-worse/, 2016. Accessed: 05.09.2022.
- [15] ENISA, Cyber Security and Resilience of smart cars Good practices and recommendations. ENISA EU, 2016.
- [16] D.-H. Lee and I.-Y. Lee, "A Lightweight Authentication and Key Agreement Schemes for IoT Environments," *Sensors*, vol. 20, 2020.
- [17] F. Basic, C. Steger, and R. Kofler, "Poster: Establishing Dynamic Secure Sessions for Intra-Vehicle Communication Using Implicit Certificates," in ACM EWSN 2022 Conf., oct 2022. poster session.
- [18] D. R. L. Brown *et al.*, "Security of ECQV-Certified ECDSA Against Passive Adversaries." Cryptology ePrint Archive, Paper 2009/620, 2009.
- [19] A. Fuchs et al., "Securing Electric Vehicle Charging Systems Through Component Binding," in Computer Safety, Reliability, and Security, pp. 387–401, 2020.
- [20] ISO 15765-2:2016, "Road vehicles Diagnostic communication over Controller Area Network (DoCAN) — Part 2: Transport protocol and network layer services," Standard, ISO, 2016.