Establishing Dynamic Secure Sessions for ECQV Implicit Certificates in Embedded Systems

Fikret Basic
Institute of Technical Informatics
Graz University of Technology
Graz, Austria
basic@tugraz.at

Christian Steger
Institute of Technical Informatics
Graz University of Technology
Graz, Austria
steger@tugraz.at

Robert Kofler
R&D Battery Management Systems
NXP Semiconductors Austria GmbH Co & KG
Gratkorn, Austria
robert.kofler@nxp.com

***Abstract*—Implicit certificates are gaining ever more prominence in constrained embedded devices, in both the internet of things (IoT) and automotive domains. They present a resource-efficient security solution against common threat concerns. The computational requirements are not the main issue anymore, with the focus now shifting to 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 as keys are tied to the certificates rather than the communication sessions. We aim to close this gap and present 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 we are able to mitigate many session-related security vulnerabilities that would otherwise remain open with only a slight computational increase of 20% compared to a static elliptic curve digital signature algorithm (ECDSA) key derivation.**

***Index Terms*—ECQV, implicit, certificate, STS, dynamic, session, key derivation, embedded, security, IoT, automotive.**

I. Introduction

Security has become 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 that only allow for a limited performance overhead [1]. In contrast, implicit certificates are a promising replacement for traditional security architecture schemes. Implicit certificates offer a lightweight certificate format, as well as a flexible public key derivation and authentication mechanism that make the use of public key infrastructures more suitable for constrained embedded systems [2]–[6].

Different schemes exist based on implicit certificates, such as Elliptic Curve Qu-Vanstone (ECQV), which is still the most popular and researched one [7]. Although numerous works have focused on ECQV and its use within 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 requires dynamic KD and it considers the state where each newly derived key has a high-enough entropy and is independent of the previous key [11]. This is especially important in session communication, where interactions happen on a frequent basis. We believe that this characteristic is often neglected due to a believed premise of the necessity for sacrificing security strength for performance gain on the limited embedded devices. As a result, static KD is often employed where key computations are directly linked to their certificate material. These keys are only updated by the change of the certificates and through re-initiating the authentication and session establishment steps. Therefore, it is 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 when the session key is compromised (e.g., via a node capturing attack by compromising a valid host device), all the captured exchanged messages could also be decrypted. This can result from any active attack that can compromise the stored device credentials and 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]. As for passive vulnerabilities, in 2018, OWASP rated for the internet of things (IoT) weak, guessable and hard-coded passwords as the number one weakness for 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]. Also, the ENISA initiative, targeted at investigating automotive security vulnerabilities, listed remote attacks, theft, and malicious surveillance as one of the most potent attacks that can happen due to the lack of 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].

978-3-9819263-7-8/DATE23/© 2023 EDAA

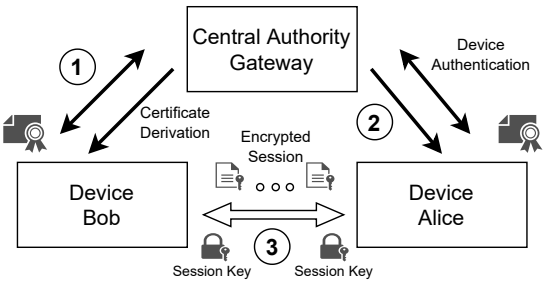

Fig. 1. Centralized implicit certificate architecture.

Our contributions. To mitigate these security vulnerabilities, we provide a solution that is independent of the rate of the certificate updates, and which ensures that each new communication session always yields a new key derivation. Additionally, we want to make sure that a session key compromise does not lead to the exposure of previous or further keys, i.e., to guarantee perfect forward secrecy. Toward this goal, 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 on embedded hardware by implementing and evaluating it on different devices, and compare it with other related implicit certificate schemes. Summarized, our main contributions in this work are:

1) Design and implementation of a dynamic key derivation approach for implicit certificate architecture schemes using the STS protocol.
2) 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 a centralized implicit certificate security architecture with three main deployment phases, 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, but it generally contains a strong central certificate authority (CA) device. The certificate derivation phase is based on ECQV and is almost identical among different solutions [3], [5], [6], [8]. The session establishment process often differs and depends on the KD and device authentication algorithms.

A. Key derivation for secure sessions

We differentiate between two sessions, the certificate session, and the communication session. The certificate session defines the duration period in which currently issued certificates are valid (e.g., in a vehicle during each new engine start). The communication session is considered as the duration in which two or multiple devices perform a single message exchange (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 presented in this work, fulfills the condition that a new session key is derived when a new communication session starts, 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 ECQV and 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 wireless networks, where the communication between the nodes is done using an SKD. For authentication, the protocol uses a Message Authentication Code (MAC) function with pre-embedded keys, but also requires the nodes to possess each others' authentication keys. A different authentication scheme is presented by Siddhartha et al. [6], where an "authenticator" is used. The authenticator consists of certificate-related data and is signed by the CA. A hash function is also used for an additional integrity check. However, the session key calculation is still based on the standard SKD.

D. Lee and I. Lee [16] present two approaches for 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 operation does not guarantee the authenticity of the device itself. The certificates and the ID could be spoofed, resulting in false identification of 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. The second approach does provide DKD and ephemeral keys. Nonetheless, both methods suffer from a central problem, the lack of device authentication, as they consider only the implicit public key calculation validity.

Recently, Zi-Yuan Liu et al. [10] presented an extension of the ECQV for use cases where devices might contain multiple certificates and keys, and offers a solution for their administration. While novel, the paper is currently not relevant for this work's use cases, as the focus of the paper is placed on larger dynamic networks prone to configuration changes.

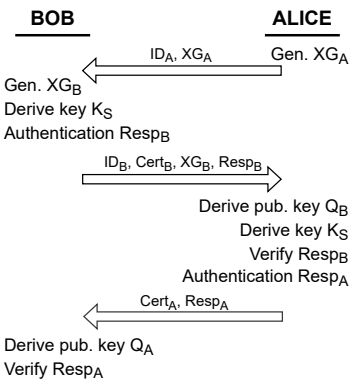

Fig. 2. Key derivation using STS protocol for ECQV architectures.

IV. A Novel Dynamic Key Derivation for ECQV

A. Security requirements

We define security requirements, intended to provide a design that answers 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. To avoid the latter, each unique key needs to have a high-enough entropy, and that is only stored, and is 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 applicable on 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 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. Compared to other STS protocol derivations, it relies on the ECQV properties and on the implicit derivation of the public key for signature verification. The security of the ECDSA with the ECQV scheme has been proven secure against passive attacks [18]. The public key calculation used for verification is derived from a certificate as:

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

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

$$X \in_R [1, ..., n-1] \rightarrow XG = X * G \quad (2)$$

The derivation of session keys K_S is done by calculating:

$$K_{PM} = X_A * XG_B = X_B * XG_A \quad (3)$$

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

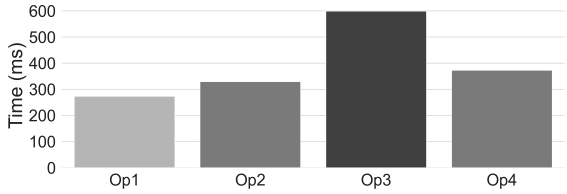

Fig. 3. Time duration of individual STS operation runs on an STM32F767.

Algorithm 1: STS implicit certificate auth. response.

Input: XG_A, XG_B, K_S
Output: $Resp$
if $device_A$ **then**
  $d_{sign} \leftarrow sign(Prk_A, (XG_A || XG_B))$
else
  $d_{sign} \leftarrow sign(Prk_B, (XG_B || XG_A))$
end
$Resp \leftarrow encrypt(K_S, d_{sign})$
return $Resp$

Algorithm 2: STS implicit certificate sign. verification.

Input: $Resp_X$, $Cert_X$
Output: $Status_{Ok}$, $Status_{Err}$
$d_{signX} \leftarrow decrypt(K_S, Resp_X)$
$Q_X \leftarrow hash(Cert_X) * decode(Cert_X) + Q_{CA}$
$Status \leftarrow verify(Q_X, d_{signX})$
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-E), the main drawback is its timely execution. Since 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. First, similar to the work presented by Sciancalepore et al. [4], the initial request can consist of both the certificate and the XG data (from Op1), while the calculations of the public key and premaster secret data (see Op2) are done in parallel. Second, it is also possible to include the subsequent Op3 to execute in parallel following Op2. However, there is a drawback 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. Even so, the actual implementation does not suffer in terms of general security since the calculations are still processed in the same manner. The main optimization advantage lies in the system design itself, which allows additional operations to run in parallel. The transmitted data in both optimizations is identical to the original protocol, but the message and content order vary slightly. Figure 3. shows operation time requirements.

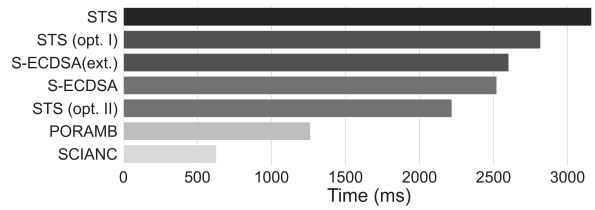

Fig. 4. Comparison of the total KD protocols processing time.

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

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

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:

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

Based on Eq. 6, no additional time is taken per device A (or B, as it is symmetrical) if the devices are identical. If they are not identical, the extra amount of time depends on the difference in their execution time for Op2 and Op3.

Ideally, if the devices are equal, the total minimal run time, derived from Eq. 5, for the two stages of optimization based on the system requirements would result in the following formulas:

$$\text{Opt. I} \quad \tau_T^{'} = 2 * T_{Op1} + T_{Op2} + 2 * T_{Op3} + 2 * T_{Op4} \tag{7}$$

$$\text{Opt. II} \quad \tau_T^{''} = 2 * T_{Op1} + T_{Op2} + T_{Op3} + 2 * T_{Op4} \tag{8}$$

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-B, 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 implementation setup

For the functional verification, we implement and run the protocols on 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 size for the SHA and HMAC, and 128-bits for the AES and CMAC. The measurements in the following sections are done using devices' system ticks and Nordic PPK2, and give an average processing time of ten runs.

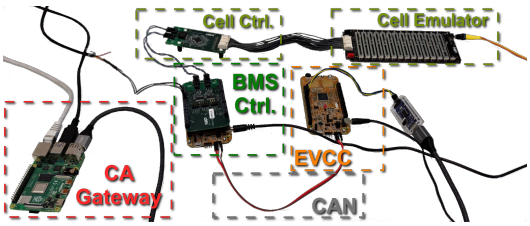

Fig. 5. Test suite for the ECQV and KD protocol evaluation.

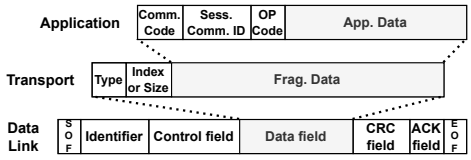

Fig. 6. CAN-FD network layers used for the session test communication.

B. 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 test four different implicit cert. KD protocols. The protocols are derived from two groups based on the used 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. The S-ECDSA is based on the traditional Diffie-Hellman with ECDSA key agreement but relies on the ECQV properties. We also consider an S-ECDSA extension, by adding an additional authentication of the acknowledgement (Ack) messages based on the *finished message* handling as seen by Porambage et al. [3]. Furthermore, we also evaluate the STS protocol when considering the optimization steps explained in Section IV-C. Note that only STS uses true DKD, while the rest fall into the SKD category. Figure 4. shows the results for the STM32F767. The results of the evaluation for all devices are shown in Table I. The run time scalability is relatively consistent regarding the devices' computational power. While STS shows the highest run time, its optimization variants have similar or faster times 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-E.

C. Overhead examination

To give a clearer analysis of the protocol processing time, it would be beneficial to consider the transmission overhead. However, this parameter is heavily dependent on the used communication protocol and its configuration. Instead, we provide an overview of the overhead for each protocol from Section V-B during one session, independent of the communication technology. We consider only the protocol-affiliated transmission data on the application level. The security algorithms' bit sizes are defined in Section V-A. We assume IDs to have a length of 16 bytes and use the minimal certificate encoding with a length of 101 bytes [7]. The results are shown in Table II.

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 ± 0.18	2894.1 ± 9.83	2521.77 ± 5.87	18.76 ± 0.11
S-ECDSA (ext.)	36882.64 ± 0.23	2976.2 ± 11.56	2602.69 ± 8.61	18.68 ± 0.12
STS	46262.03 ± 0.13	3622.71 ± 7.034	3162.07 ± 7.52	23.26 ± 0.12
STS (opt. I)	41680.23 ± 1.2	3246.55 ± 12.97	2818.02 ± 11.26	20.87 ± 0.07
STS (opt. II)	32410.81 ± 1.14	2556.84 ± 13.13	2219.25 ± 11.3	16.31 ± 0.07
SCIANC	8990.49 ± 0.03	721.67 ± 0.28	628.1 ± 0.32	4.58 ± 0.02
PORAMB	17932.17 ± 0.05	1471.66 ± 0.63	1263.0 ± 0.42	8.98 ± 0.04

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)	A1: ID(16), XG(64)	A1: ID(16), Nonce(32), Cert(101)	A1: Hello(32), ID(16)
	B1: ID(16), Cert(101), Sign(64), Nonce(32)	B1: ID(16), Cert(101), XG(64), Resp(64)	B1: ID(16), Nonce(32), Cert(101)	B1: Hello(32), ID(16)
	A2: Cert(101), Sign(64)	A2: Cert(101), Resp(64)	A2: Auth_MAC(32)	A2: Cert(101), Nonce(32), MAC(32)
	B2: ACK(1), (+Ext_Fin(96))	B2: ACK(1)	B2: Auth_MAC(32)	B2: Cert(101), Nonce(32), MAC(32)
	A3: (+Ext_Fin(96))			A3 & B3: Finish(197)
Total	4(+1): 427(+192) B	4: 491 B	4: 362 B	6: 820 B

Both the S-ECDSA and STS protocols show similar transmission sizes, and also the least communication steps if the last Ack message is not considered. The SCIANC protocol also requires only four transmissions, but with only 362 bytes total in the assumed setup. Contrarily, the PORAMB algorithm shows the largest overhead with 6 total steps and 820 bytes. We do not include the optimized version of STS since it does not differ from the original 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 is minimal in comparison to the individual KD protocols. This is further complemented by the prototype evaluation results from Section V-D.

D. Prototype implementation evaluation

In order to evaluate the proposed protocol design w.r.t. its technical use, we implement a prototype system that depicts a common communication occurrence between two ECUs in an automotive network. The prototype 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 NXP Semiconductors to portray a real-world environment. The BMS is additionally connected to a battery cell controller and a battery emulator to emulate 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 a CAN-FD derivation with an implemented CAN-TP layer for message fragmentation [20]. Figure 6 shows the

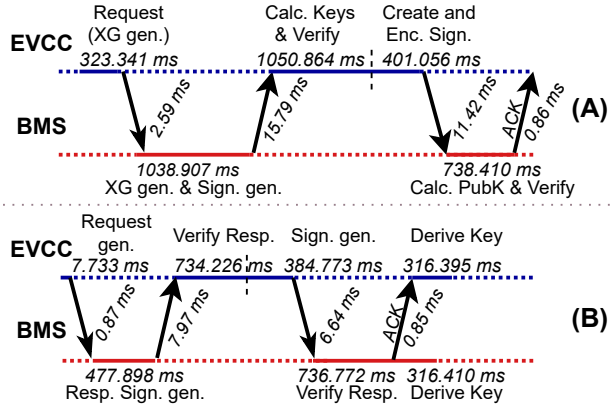

Fig. 7. Timeline model of the prototype session communication between a BMS and EVCC for: (A) STS & (B) S-ECDSA, ECQV KD protocols.

message formats. The devices also communicate with a more powerful CA gateway (i.e., a Raspberry Pi 4) to handle the initial device authentication and certificate distribution. The CAN-FD nominal phase bit rate is configured to 0.5 Mbit/s, and the data phase rate is 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 do 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

TABLE III
SECURITY OVERVIEW OF THE KD PROTOCOLS FOR ECQV.

	S-ECDSA	STS	SCIANC	PORAMB
Data exposure	X	✓	X	X
Node capturing	Δ	Δ	X	X
Key data reuse	X	✓	Δ	X
Key der. exploit	Δ	✓	Δ	Δ
Auth. procedure	✓	✓	Δ	Δ

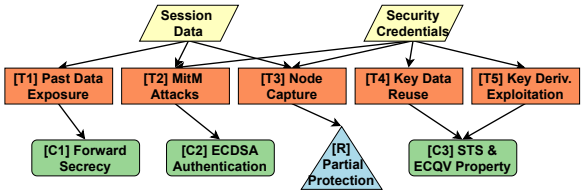

Fig. 8. Block diagram representation for the STS-ECQV KD threat model.

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

E. Security analysis

In this section, we analyse the listed threats from Section IV-A and compare the previous KD protocols on the provided security level. We also specially look at the mutual authentication procedure, as it is an important feature against MitM attacks. The results of the analysis are presented in Table III, with the following notation: X - weak or no countermeasure, Δ - partial protection, ✓ - 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, but not for 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 to 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. We have further compared the model to other KD protocols on embedded devices and discussed their properties. The STS offers a good balance between providing additional security features and certainty without compromising much of the performance. Our experimental evaluation has shown a slight run time increase of only $\approx 21\%$ compared to a static ECDSA KD protocol, without additional communication overhead. While other non-EC authentication-based KD protocols have shown noticeably faster execution times, they lack the security level acceptable for modern systems. To compensate for the STS run time, we introduce a series of optimization steps for the protocol operations. In future work, we plan to investigate the influence of security modules and hardware accelerators when considering the implicit certificate protocols on embedded devices.

ACKNOWLEDGMENT

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

REFERENCES

[1] 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 *Proc. of the 23rd IEEE ICIT Conf.*, 2022. In Press.
[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] SEC Consult, "House Of Keys: 9 Months Later… 40% Worse." https://sec-consult.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.