Lightweight Offline Access Control for Smart Cars

Gian-Luca Frei
Zühlke Engineering AG
Bern, Switzerland
emails: hello@gianlucafrei.ch
gifr@zuehlke.com

Fedor Gamper
Swiss Federal Railways
Bern, Switzerland
email:
fedorgamper@outlook.com

Prof. Dr. Annett Laube
Bern University of Applied Sciences TI -ICTM
Biel/Bienne, Switzerland
email:
annett.laube@bfh.ch

Abstract—**In this paper, a novel access control protocol that offers appealing features for carsharing is presented. It describes how a user can authenticate and authorize himself using a smartphone on an immobilizer in a car. First, it requires no online connection to open cars. Therefore, it is suitable for applications where the cars and the users have no network connection. Second, the protocol is designed for low-bandwidth channels like Bluetooth Low Energy and transports around 210 bytes per car access. Third, it enables users to delegate their access rights to other users. These properties were achieved by using custom public key certificates and authorization tokens with a public key recovery mechanism.**

***Keywords*– access control; authentication; authorization; bluetooth low energy; carsharing; cryptographic protocol; public-key cryptography; public-key recovery.**

I. INTRODUCTION

Smartphones have become omnipresent devices. At the same time, the worldwide market for carsharing has grown exponentially over the last decade [1] [2]. As a result, many carsharing providers offer their customers the possibility to open rental cars with smartphones. Often, the security of such systems is unknown because the vendors keep the system design secret.

Designing access control solutions is quite easy if the cars have a stable network connection. The only parts needed are an authentication mechanism and a server that the car can query to check if a user is allowed to access it. However, maintaining a constant network connection is often not possible or not desirable because of the higher costs involved. Moreover, developing a protocol whereby all steps can be done with no network connection would not be meaningful in a world where most smartphones almost always have an internet connection. Therefore, we use the following networking model: The users are most of the time online and need a network connection for registration and to make bookings. Later, a user receives a credential that enables him to make use of his access rights in an offline fashion. This means that if he opens a car, he needs no network connection, nor does the car need a network connection. To illustrate this, imagine the car is located in an underground car park, where no cellular reception is available. In that case, the carsharing provider cannot communicate with the car, whereas a user can cross into the offline zone by entering the underground car park. Most carsharing providers allow their users to make spontaneous bookings over their smartphones. This means access control rules can change quickly. Therefore, the user needs to receive the credentials to authenticate and authorize himself outside before entering the underground car park.

The most convenient way to establish communication between a car and a smartphone is through either Bluetooth or Near-Field Communication (NFC) [3]–[5]. Bluetooth Low Energy (BLE) is part of the Bluetooth 4 specification and is designed to use little electric power [6]. Another advantage of Bluetooth Low Energy is that no device-pairing is needed. This makes Bluetooth Low Energy very convenient. NFC is also very convenient but is not fully supported on Apple devices [7]. This makes BLE a popular choice for real-world applications. A downside of BLE and NFC are that the transmission speed is low and often the theoretical bandwidth cannot be reached in practice. In our tests with Bluetooth LE, we measured a transmission speed of under 1,000 bytes per second [8]. It is, therefore, important to keep the sizes of the messages exchanged between the smartphone and the object as small as possible because large messages can have direct negative impacts on usability.

This paper is organized as follows: Section II presents the current state-of-the-art of access control protocols for carsharing. Then, in Section III the new protocol is presented. Section IV discusses possible ways to attack a system that uses the presented protocol. Finally, Section V concludes the paper.

II. STATE-OF-THE-ART

This section gives an overview of the existing work on carsharing systems. Dmitrienko et al. presented an offline access control system for free-float car sharing. This protocol is based on symmetric encryption, secure elements to store private credentials and a single carsharing provider that manages access rights [9]. Dmitrienko et al. also proposed a generic access control system based on NFC enabled devices which also supports offline validation and delegation of authorization. However, this work makes use of some proprietary protocols [10]. SePCAR is an access control protocol for smart cars. However, the focus of this protocol is more on user privacy than on bandwidth efficiency [11]. Mustafa et al. published a comprehensive requirements analysis for carsharing systems [12]. There exist also protocols that are not intended for carsharing but could also be used in this context. Grey is a research project which has been used to access physical space, computer logins and web applications based on asymmetric crypto on smartphones [13]. With Grey, users can pass their authorizations to other users. Arnosti et al. proposed a general physical access control system that uses NFC to communicate with digital and physical resources. However, their protocol requires a network connection between the resource and a central server [14]. Groza et al. explored the use of trusted platform modules along with identity-based signatures for vehicle access-control [15]. Ouaddah et al. developed an access control framework for internet of things applications based on blockchain technology [16]. Similar to public key recovery which is used in the presented protocol is another technique called implicit certificates. Some IoT-Protocols make use of this technique, example are developed by Sciancalepore et al. and Ha et al. [17] [18]. Furthermore, there are many proprietary solutions mostly from carsharing companies, but no details are publicly available. Examples are from Zipcar, OTA-Keys, Continental Cars, and Valeo. To the best of our knowledge no prior work has focused on low bandwidth protocols for access control in carsharing. To fill this research gap, we propose a new lightweight access control protocol for offline cars.

III. Protocol

In this section, the developed access control protocol is presented.

A. Overview

The protocol is based on the principle of authorization tokens and a strong authentication mechanism with public key certificates. Each user has a device containing a unique private key used for authentication and a public key certificate. Further, each user has one or multiple authentication tokens, which are digitally signed messages that link access rights to a specific user. These tokens are independent of the private key used for authentication and can be shared between multiple devices of one user. For example, if a user changes his smartphone, then he needs to onboard his new phone to generate a new private key and get a new public key certificate and then he can copy his authorization tokens to his new device. The user can then create an access request on his device. Each access request is authenticated with the private key and linked to the user with the public key certificate. Authentication with public key certificates is a widely used and secure authentication mechanism and removes many attack points because outsiders can forge access requests of regular users only with negligible probability. The most important advantage of an authorization token is that the car only needs to store a few public keys as a trust anchor. It uses these saved public keys to check access requests. This is useful for applications where the car is offline for a long time or the hardware of the car needs to be inexpensive.

a) Components and Roles: A user can access a car from different devices. A user device can be a smartphone, a smart card, or a computer. To use a new device, the user needs to introduce it to the system by performing the onboarding process with it. During device onboarding, a new private key is generated and an Identity Authority (IA) issues a public key certificate for the new device. The car needs to have a computing platform that communicates with the user device, validates the access requests and controls the immobilizer of the car. A car owner is a person who has administrative access to a car and configures its computing platform. There are two different authority roles. The IA checks the identity of users and issues public key certificates for new user devices. The Permission Authority (PA) issues authorization tokens. The cars can trust multiple authorities. Figure 1 visualizes the relationships between the different components. One party can be an IA and PA at the same time; however, the two roles can also be split between different parties. For instance, in peer-to-peer sharing, each car owner could trust a car-sharing platform to check identities and driving licenses but run a PA by himself. The protocol does not specify a user registration method. We assume that identity authorities have a way to manage users and that users can authenticate to identity authorities. The public key certificates are only used to authenticate a user to a car.

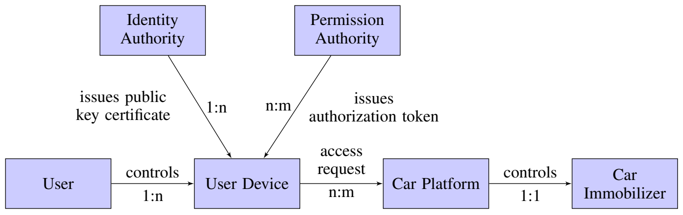


Figure 1. Components and Roles

b) Basic Description: Each user in the system owns an asymmetric key pair used for authentication. During the onboarding process, an Identity Authority signs a public key certificate for the user. Next, the user receives an authorization token which is signed by the Permission Authority. To access a car, the user signs an access request which contains a description of what he wants to do. This access request message, together with a certificate and authorization token, is then sent to the car. The car verifies the access request. If the car trusts all the involved authorities and the request is valid, it grants access. The car trusts a set of authorities by storing their public keys in a local trust store. An authorized user can delegate a subset of his access rights to another user. For instance, when a user booked a car, he might not use the car only by himself but wants that his travel companion is able to open the car too. To realize token delegation, all authorization tokens have a flag that indicates whether the authorized user is allowed to delegate his access rights to another user. To delegate an access right, an authorized user A signs a new token for another user B. User B then uses the chain of tokens, containing his token and the token of A, to claim or further delegate his access rights.

c) Cryptographic Primitives: The cryptographic primitives used are a cryptographic hash function $H(m)$ and the Elliptic Curve Digital Signature Algorithm (ECDSA) with public key recovery. $recoverPk(h, s)$ is a function which computes a public key pk which is valid for the digest h and signature s. ECDSA is one of the only digital signature schemes where this operation is possible [19] [20] [8].

B. Protocol Phases

In this section, we describe the different phases of the protocol. Generally, if any check fails, the process must be aborted. Figures 2 to 7 illustrate the different processes.

1) Car Initialization: (Figure 2) During the initialization process, a car owner CO adds a new car to the system by setting up the car platform CP. First, the CO sends the $SystemParameters$ consisting of the digital signature algorithm and the hash function to the CP. Next, he sends the set of trusted public keys PK_{IA}, PK_{PA} on the CP. These public keys are later used to check the authenticity of public key certificates and authorization tokens. Further, the CO should check if the clock to the CP is precise enough. How precise the clock must be can vary between different systems; however, the divergence should usually not exceed a few seconds. A precise time source is necessary because the car needs the current time to check if the access tokens are valid at the moment of usage.

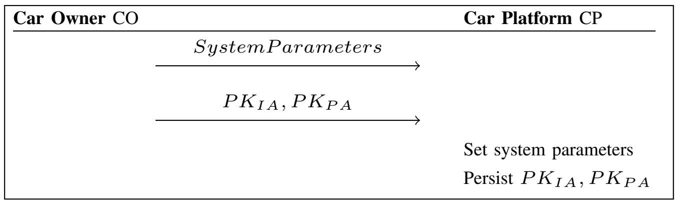


Figure 2. Car Initialization Sequence Diagram

2) Device Onboarding: (Figure 3) During onboarding, a user U sets up a new personal device D to later access a car. The device usually communicates with the authorities over an internet connection; therefore, minimal message sizes are not important during this process. The device needs to generate a new private key. Then it requests a public key certificate from an IA for the public key which belongs to the private key. This process needs a mutually authenticated channel between D and IA. D first generates a key pair (sk, pk) where sk is the platforms new private key and pk is the corresponding public key. D then generates a proof of knowledge of sk by signing $H(u, pk)$ with sk. Then it sends this signature s together with is username u and public key pk to the IA. This proof of knowledge is needed to prevent a user from getting a certificate for a key pair without the knowledge of the private key. The IA then checks this proof and then signs $s_C = sign_{skIA}(H(u, pk, v))$ where v is the validity period of the public key certificate and sends the certificate $cert = (u, v, s_C)$ to D. Note that the public key is hashed into the signed part of the certificate but not contained in the certificate itself. The public key can later be recovered and then the authenticity of the certificate can be checked.

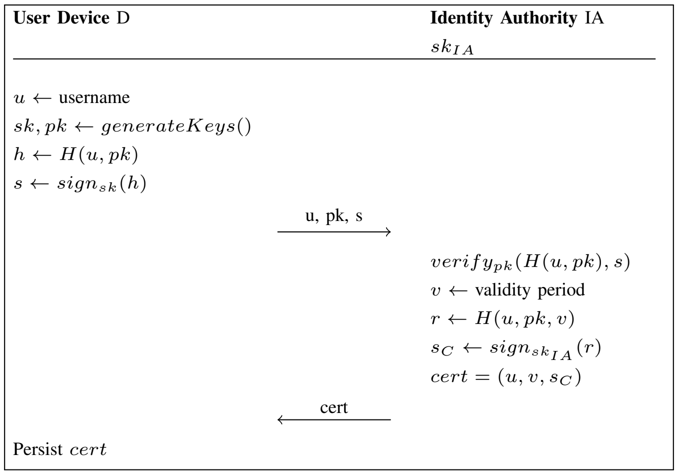

User Device D
Identity Authority IA
sk_{IA}
$u \leftarrow$ username
$sk, pk \leftarrow generateKeys()$
$h \leftarrow H(u, pk)$
$s \leftarrow sign_{sk}(h)$
u, pk, s
$verify_{pk}(H(u, pk), s)$
$v \leftarrow$ validity period
$r \leftarrow H(u, pk, v)$
$s_C \leftarrow sign_{sk_{IA}}(r)$
$cert = (u, v, s_C)$
cert
Persist $cert$

Figure 3. Device Onboarding Sequence Diagram

3) Root Token Issuance: (Figure 4) The result of the root token issuance process is an authorization token t. This token allows the user to claim access rights to a car or to delegate his rights to another user. If a token t_n is delegated, all tokens used $t_1, ..., t_{n-2}, t_{n-1}$ to delegate the last token t_n are needed to check the validity of the last delegated token. This sequence of tokens $t_1 - t_n$ is called a chain of tokens T. The first token t_1 in such a chain needs to be issued by a PA and is called the root token. To issue a root token, a PA signs an authorization token t and sends this token to the device D of the user. How the PA manages the access rules depends on the application of the protocol and is not specified. The token t is a signed message consisting of p, which is a description of the access rights of the user. It also contains a bit-flag d, which indicates if the user can delegate his access rights and a signature $s_{T1} = sign_{sk_{PA}}(H(u, p, d))$. To keep the protocol flexible, the content of p is not specified. It should contain a set of cars that the user may access and a validity timespan. The root token then is $t_1 = (p, d, s_{T1})$. The PA sends the sequence T_1 to the D of the user. D stores T_1 together with $C_1 = (cert)$ which is the sequence of the corresponding public key certificates. Note that the username u itself is part of the signed hash. However, it is not part of the token message like the public keys in certificates. The reason for this is that an authorization token must always be checked with the corresponding public key certificate containing the same username. Instead of testing if both usernames are the same, u can be taken from the public key certificate and so u can be omitted from the authorization token.

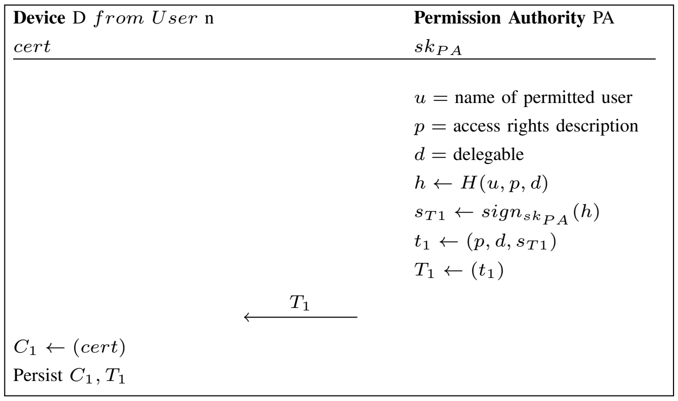

Device D $from\ User$ n
$cert$
Permission Authority PA
sk_{PA}
$u =$ name of permitted user
$p =$ access rights description
$d =$ delegable
$h \leftarrow H(u, p, d)$
$s_{T1} \leftarrow sign_{sk_{PA}}(h)$
$t_1 \leftarrow (p, d, s_{T1})$
$T_1 \leftarrow (t_1)$
T_1
$C_1 \leftarrow (cert)$
Persist C_1, T_1

Figure 4. Root Token Issuance Sequence Diagram

4) Token Delegation: (Figure 5) To delegate a token, the delegating user with username u_{n-1} enters the name of the receiver u_n, the description of the rights he wants to delegate p_n and the flag d_n, which indicates if the new token can further be delegated to his device D. D then sign the new token t_n with the private key sk_{n-1} in the same way as in the root token issuance process, except that in the new sequence of tokens T_n, t_n is appended to the prior sequence of tokens T_{n-1}. D sends the new chain of tokens T_n and the prior chain of certificates C_{n-1} to the device of the receiver D'. D' appends his public key certificate to the chain of certificates C_{n-1}. The receiving user can further delegate his token, if the received token is delegable, by performing this process again.

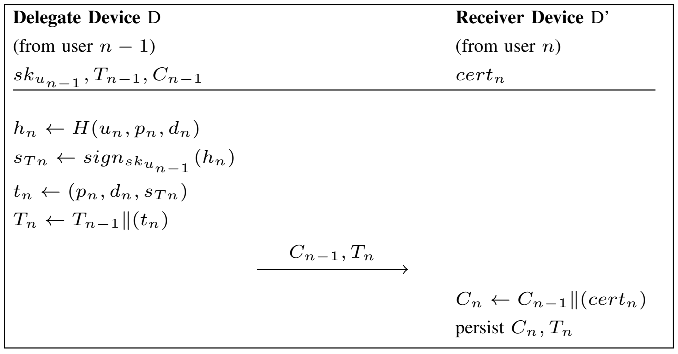

Delegate Device D
(from user $n-1$)
$sk_{u_{n-1}}, T_{n-1}, C_{n-1}$
Receiver Device D'
(from user n)
$cert_n$
$h_n \leftarrow H(u_n, p_n, d_n)$
$s_{Tn} \leftarrow sign_{sk_{u_{n-1}}}(h_n)$
$t_n \leftarrow (p_n, d_n, s_{Tn})$
$T_n \leftarrow T_{n-1} \| (t_n)$
C_{n-1}, T_n
$C_n \leftarrow C_{n-1} \| (cert_n)$
persist C_n, T_n

Figure 5. Token Delegation Sequence Diagram

5) Access Request: (Figure 7) To create an access request, the user needs a list of tokens $T = t_1, \dots, t_{i-1}, t_i$ and a list of certificates C. Each t_i in T except t_1 must be signed by the public key corresponding to the certificate c_{i-1}. The root token t_1 must be signed by a Permission Authority. Also, the user of c_i and t_i must be the same. Figure 6 illustrates a chain of tokens and the corresponding certificates resulting from two delegations. The fields in parentheses are hashed into the signature but are not stored in the message. To access a car, the user U enters a description of the access request $desc$ into his device D. This is needed in case a car allows different accesses or needs additional information. For instance, the description can specify that the car should open the trunk only. For this description, together with, the current time τ and the name of the car r, D creates a signature $s = sign_{sk}(H(desc, \tau, r))$. Finally, τ, the list of tokens T, the list of certificates C, the description $desc$ and s are sent to the car platform CP. CP validates the chain of permissions according to the $validateRequest$ procedure (Figures 8 and 9) and if no check fails, it grants accesses to the car according to the description $desc$. The authentication mechanism depends on the hardness to forge the digital signature s. For an attacker, without the knowledge of the private key sk that belongs to the public key certificate, it is not realistic to forge a valid signature s. An attacker could record and try to replay an access request. To prevent this, the timestamp τ is also included in the signed part of the request. The car must check if the timestamp is close to the current time provided by the clock of the car platform. How small the derivation can be is a trade-off between susceptibility for timing errors and security.

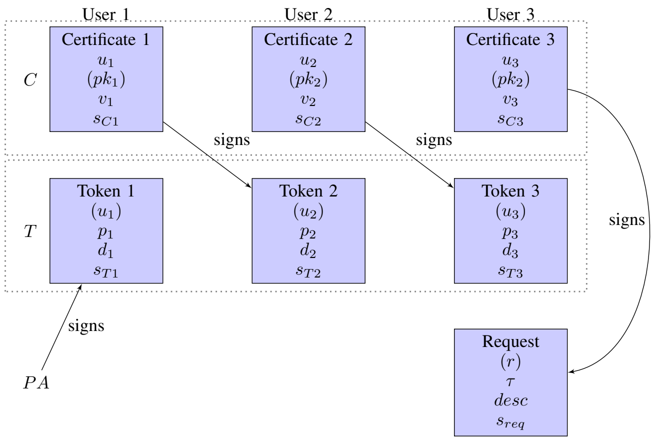
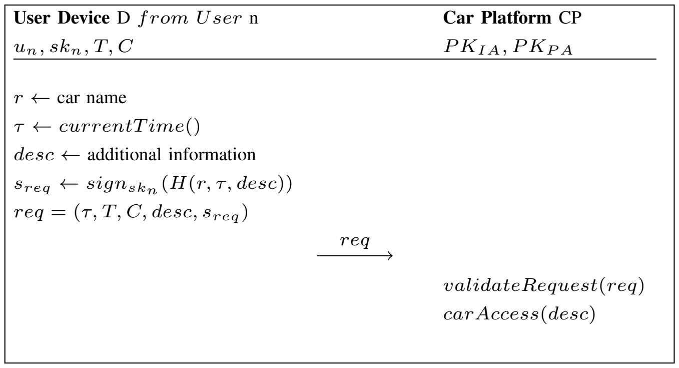


Figure 6. Access Request Example

C. Analysis of the Access Request Size

In this section, the size of the access request message is analyzed. This size is very important because the channel between the user device and car platform often has a low bandwidth.

a) Security Parameter: When implementing an application of this protocol, a security parameter S has to be chosen. This parameter is a way to define how difficult it should be for an attacker to break the cryptographic primitives of the application. More precisely: A polynomial bound attacker is expected to break the primitives in $O(2^S)$ computing steps. Nowadays a security parameter of about 112 is recommended to protect secrets for about 10 years [21] [22]. However, this protocol only provides authentication and authorization. It does not provide confidentiality. The keys used to sign the messages are only valid for a specific time and should then be renewed. An attacker is not interested in breaking old keys. Therefore, a smaller security parameter is also possible when all keys (especially the authority keys) are regularly replaced by freshly generated keys. To achieve a security level of S a $2S$ bit curve must be used (for instance a 256 bit curve for 128 bit security).


User Device D $from\ User$ n — **Car Platform** CP

u_n, sk_n, T, C — PK_{IA}, PK_{PA}

```
$r \leftarrow$ car name
$\tau \leftarrow currentTime()$
$desc \leftarrow$ additional information
$s_{req} \leftarrow sign_{sk_n}(H(r, \tau, desc))$
$req = (\tau, T, C, desc, s_{req})$
                    $\xrightarrow{req}$
                                        $validateRequest(req)$
                                        $carAccess(desc)$
```

Figure 7. Access Request Sequence Diagram

```
$validateRequest(\tau, T, C, desc, s_{req}, PK_{IA}, PK_{PA})$
$n \leftarrow \mathsf{length}(C)$

/ Authentication Check
check $currentTime()$ is near $\tau$
$r \leftarrow$ car name
$h \leftarrow H(r, \tau, desc)$
/ $c_n^u$ means the username of the nth certificate
$signatureCheck(h, s_{req}, c_n, c_n^u, PK_{IA})$

/ Check if the access description is within the claimed access privileges
/ $t_n^p$ means the privileges of the nth token
Check $desc$ is within $t_n^p$
Check context of $t_n^p$

/ Check the root token
/ $t_i^s$ means the signature of the ith token
$h \leftarrow H(c_1^u, t_1^p, t_1^d)$
$pk_{PA} \leftarrow recoverPk(h, t_1^s)$
Check $pk_{PA} \in PK_{PA}$

/ Check the chain of tokens
for $i \leftarrow 2, 3, ..., n$
    $h \leftarrow H(c_i^u, t_i^p, t_i^d)$
    $signatureCheck(h, t_i^s, c_{i-1}, c_{i-1}^u, PK_{IA})$
    / $t_n^p$ means the privileges of the nth token
    Check $t_i^p$are within $t_{i-1}^p$
    Check $t_i^u = c_i^u$
    Check $t_{i-1}^d = 1$
```

Figure 8. Validate Request Procedure

```
$signatureCheck(h, s, cert, u, PK_{IA})$
$pk \leftarrow recoverPk(h, s)$
$r \leftarrow H(u, pk, cert_v)$
$pk_{IA} \leftarrow recoverPk(r, cert_s)$
Check $pk_{IA} \in PK_{IA}$
Check $currentTime()$ within $cert_v$
```

Figure 9. Signature Check Procedure

An ECDSA signature then has a size of $4S$ plus two bits for the recovery parameter, which are negligible for our calculations.

b) Size of an Access Request: An access request consists of at least one certificate and one token, each containing one digital signature. Additionally, we have one signature for the access request message itself. For each delegation, we need to add one certificate and one token. Let $s = 4S$ be the size of a digital signature. The total size of the access request is therefore $s + a + (d + 1)(2s + b)$ where d is the number of delegations. a is the size of the fields in the request (τ, $desc$), and b is the size of the fields in the certificate and token (username, validity, privileges). Because a and b are relatively small, we assume them to be $a \approx 20$ bytes and $b \approx 40$ bytes. Table I shows the calculated access request message sizes.

TABLE I. CALCULATED SIZE OF THE ACCESS REQUEST

		for S=100
No delegation:	$3s + a + b \approx 3s + 60B$	$\approx$ 210 bytes
One delegation:	$5s + a + 2b \approx 5s + 100B$	$\approx$ 350 bytes
Per additional delegation:	$2s + b \approx 2s + 50B$	$\approx$ 140 bytes

D. Proof of Concept

To test the efficiency of protocol, we created a prototype which consists of a mobile application and a Raspberry Pi 3b+ running a Node.js application [23]. An iPhone X was used to transmit an access request over Bluetooth Low Energy (BLE) to the raspberry witch was simulating the car. It took in average 495 ms to transmit and compute an access request message with no delegations. This was measured in an environment with no other BLE devices nearby and 15 cm distance between the devices. With one delegation, the transmission and computation time increased to in average 566 ms and with two delegations to in average 732 ms. This value could be minimized by improving the computational performance of the prototype [8].

IV. ANALYSIS OF ATTACK VECTORS

This section discusses different vectors an adversary could use to attack a system using this protocol and how the protocol protects against such attacks.

a) Denial of Service: Communication channels over the air are in general vulnerable to a denial of service attacks. Therefore, an attacker could prevent an authorized user from accessing a car. For applications where availability is important, it is better to use a low range technology such as NFC instead of Bluetooth.

b) Man-in-the-middle: An attacker can capture a message in transit and forward it to the car, but since all part of the access request are authenticated he cannot alter it. However, if an attacker captures the request messages, it could have a privacy impact. For applications where privacy is important, it is recommended to encrypt the access requests in such a way that only the targeted car can decrypt it. Another type of attack would be when an attacker tries to extend the range of the communication channel between the user and the car. He could trick the user to unintentionally open a car. To prevent this, the user device should only send access requests when the user confirms that he is near the car.

c) Clock-Synchronization: If the time source of the car is not correct, a user with a valid token could access the car outside the validity timespan of the token. Also, the validity of the authentication certificates could be circumvented. It is therefore important that only the car owner can adjust the time source of the car. Depending on the application a small derivation of a few seconds can be unproblematic, but longer differences should be prevented.

d) Replay Attacks: An attacker can copy a transmitted access request and replay it later. However, the time stamp τ in the access request prevents the car from accepting the replayed access request because it compares τ to the current time.

e) Abuse of the Car: A malicious user who has access to a car could use it in an unintended way. For example a customer could try to manipulate the car's computing platform. To prevent this, the computing platform should be physically protected against such and similar manipulations, or at least able to detect it. Also, a user could rent a car and not return it on time. Such and similar attacks should be regulated in the general business terms of the carsharing provider.

f) Attacks on the Authentication Mechanism: An attacker could try to circumvent the authentication mechanism of the access request and impersonate another user. To do this, the attacker would need to forge a valid public key certificate of a trusted IA or forge a valid signature in the access request. Both types of attack are prevented by the difficulty of forging a digital signature.

g) Attacks on the Authorization Mechanism: An attacker which is a registered user could also try to circumvent the authorization mechanism by trying to forge a valid chain of authorization tokens. To do this, the attacker would need to forge either the signature of one authorization token or forge a public key certificate.

h) Attacks on the Devices of Other Users: An attacker could try to steal the private key of another user's device or make another user's device to sign a token or access request by installing malware on the victim's device. To prevent this, the users' device must be secured against such attacks. To make this attack more difficult, the private key should be stored on a trusted platform module with a key activation function, which most modern devices provide.

i) Attacks on the Identity Authorities: An attacker could try to attack an IA directly. An attacker that intruded into an authority could either steal the private key or make the authority to sign a public key certificate. The attacker then could impersonate any other user. Once the intrusion is detected, all car owners would need to remove the public key of the corrupted authority on all cars that trusted that authority.

To prevent such an attack, the IA must be secured against different cyberattacks to make such an attack unprofitable for an attacker.

j) Attacks on the Permission Authorities: Like the attack on the identity authorities, an attacker could also steal the private key of a PA. He then could issue arbitrary root tokens and could access all cars that trust the corrupted authority. However, to use the car, the attacker also needs to authenticate himself and therefore needs a public key certificate. This means that such an attack would only be interesting for an already registered user or in combination with another attack. Similar to the attack on the identity authorities, the PA must be secured against different cyber attacks.

To sum up, attacks on the authorities are the most promising attacks. As a result, these authorities must be well secured to mitigate such threats. Splitting up the privileges of the authorities also minimizes the impact of a successful attack.

V. Conclusion

The presented protocol enables carsharing providers to use a secure access control mechanism over Bluetooth Low Energy or Near-Field Communication. It is based on well-known security mechanisms but uses the less-known technique of public key recovery to reduce the size of the messages. The security mechanism of the protocol is based on the principle of public-key certificates and digitally-signed authorization tokens. Both mechanisms are well known and used for a wide variety of applications. The main novelty of our approach is the use of public-key recovery, to drastically reduce the message sizes of the custom certificates. The custom public key certificates we developed, have a size of only about 100 Bytes for a 256-bit key. Compared to traditional X.509 [24] or GPG [25] certificates for the same ECDSA key, this is about 5 to 10 times smaller. The networking model of this protocol assumes that only the car and the smartphone can communicate with each other during the access phase. A consequence of this model is that the revocation of an access right is not possible. If a carsharing service allows a user to open a car in places where no network connection is possible, it gives up the possibility to communicate with the car. Thus, the car cannot ask if the user's access rights have been revoked.

To the best of our knowledge, it is the first access control protocol that makes use of this technique. As a result, it has powerful and interesting properties, which makes it suitable for carsharing-applications. Nevertheless, the protocol could also be used in other domains such as building door systems. The developed prototype proves that the protocol runs fast on today's smartphones and is very convenient for the users.

References

[1] S. Le Vine, A. Zolfaghari, and J. Polak, "Carsharing: evolution, challenges and opportunities," Scientific advisory group report, vol. 22, 2014, pp. 218–229.

[2] S. Shaheen, E. Martin, and B. Apaar, "Peer-to-peer (p2p) carsharing: Understanding early markets, social dynamics, and behavioral impacts," 2018.

[3] S. International Organization for Standardization, Geneva, "Iso/iec 18092:2013," Tech. Rep., [retrieved: March, 2020]. [Online]. Available: https://www.iso.org/standard/56692.html

[4] R. Want, "Near field communication," IEEE Pervasive Computing, no. 3, 2011, pp. 4–7.

[5] K. Finkenzeller, RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication. John Wiley & Sons, 2010.

[6] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," Sensors, vol. 12, no. 9, 2012, pp. 11 734–11 753.

[7] Apple. Core nfc documentation. [retrieved: March, 2020]. [Online]. Available: https://developer.apple.com/documentation/corenfc

[8] G.-L. Frei and F. Gamper, "Design and implementation of a digital access control protocol," B.S. thesis, Bern University of Applied Science, 2019.

[9] A. Dmitrienko and C. Plappert, "Secure free-floating car sharing for offline cars," in Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. ACM, 2017, pp. 349–360.

[10] A. Dmitrienko, A.-R. Sadeghi, S. Tamrakar, and C. Wachsmann, "Smarttokens: Delegable access control with nfc-enabled smartphones," in Trust and Trustworthy Computing, S. Katzenbeisser, E. Weippl, L. J. Camp, M. Volkamer, M. Reiter, and X. Zhang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 219–238.

[11] I. Symeonidis, A. Aly, M. A. Mustafa, B. Mennink, S. Dhooghe, and B. Preneel, "Sepcar: A secure and privacy-enhancing protocol for car access provision," in Computer Security – ESORICS 2017, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds. Cham: Springer International Publishing, 2017, pp. 475–493.

[12] I. Symeonidis, M. A. Mustafa, and B. Preneel, "Keyless car sharing system: A security and privacy analysis," in 2016 IEEE International Smart Cities Conference (ISC2). IEEE, 2016, pp. 1–7.

[13] L. Bauer, S. Garriss, J. M. McCune, M. K. Reiter, J. Rouse, and P. Rutenbar, "Device-enabled authorization in the grey system," in Information Security, J. Zhou, J. Lopez, R. H. Deng, and F. Bao, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 431–445.

[14] C. Arnosti, D. Gruntz, and M. Hauri, "Secure physical access with nfc-enabled smartphones," in Proceedings of the 13th International Conference on Advances in Mobile Computing and Multimedia, ser. MoMM 2015. New York, NY, USA: ACM, 2015, pp. 140–148.

[15] B. Groza, L. Popa, and P.-S. Murvay, "Carina-car sharing with identity based access control re-enforced by tpm," in International Conference on Computer Safety, Reliability, and Security. Springer, 2019, pp. 210–222.

[16] A. Ouaddah, A. Abou Elkalam, and A. Ait Ouahman, "Fairaccess: a new blockchain-based access control framework for the internet of things," Security and Communication Networks, vol. 9, no. 18, 2016, pp. 5943–5964.

[17] S. Sciancalepore, A. Capossele, G. Piro, G. Boggia, and G. Bianchi, "Key management protocol with implicit certificates for iot systems," in Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems. ACM, 2015, pp. 37–42.

[18] D. A. Ha, K. T. Nguyen, and J. K. Zao, "Efficient authentication of resource-constrained iot devices based on ecqv implicit certificates and datagram transport layer security protocol," in Proceedings of the Seventh Symposium on Information and Communication Technology. ACM, 2016, pp. 173–179.

[19] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," International journal of information security, vol. 1, no. 1, 2001, pp. 36–63.

[20] C. Research, "Standards for efficient cryptography, SEC 1: Elliptic curve cryptography," September 2000, version 1.0.

[21] E. Barker and Q. Dang, "Nist special publication 800-57 part 1, revision 4," NIST, Tech. Rep, 2016.

[22] N. Smart et al., "Algorithms, key size and protocols report (2018)," ECRYPT—CSA, H2020-ICT-2014—Project, vol. 645421, 2018.

[23] G.-L. Frei and F. Gamper. Loac-protocol prototype. [retrieved: March, 2020]. [Online]. Available: https://github.com/gianlucafrei/LOACProtocol (2019)

[24] Microsoft. X.509 public key certificates. [retrieved: March, 2020]. [Online]. Available: https://docs.microsoft.com/en-us/windows/desktop/seccertenroll/about-x-509-public-key-certificates

[25] GnuPG. The gnu privacy guard. [retrieved: March, 2020]. [Online]. Available: https://www.gnupg.org/index.html