Implementation and Adaptation of the Pseudonymous PKI for Ubiquitous Computing for Car-2-Car Communication

Stefan Kaufmann

IT-Designers Gruppe
STZ-Softwaretechnik
Im Entennest 2
73730 Esslingen
stefan.kaufmann@stz-softwaretechnik.de

Abstract: Car-2-Car communication requires the use of pseudonymous public keys to ensure the vehicle's privacy by making its messages unlinkable over a longer period. The Pseudonymous PKI (PPKI) for Ubiquitous Computing by K. Zeng [Ze06] offers peer-side pseudonym generation, non-repudiation and an efficient revocation mechanism that excludes the revoked peers from the group of authorized participants. In addition, this work provides a new authorisation scheme that incorporates efficiently into the pseudonymous PKI, offers peer-side pseudonymous credential generation and preserves the peer's privacy during the initial authorisation obtainment from third party authentication authorities. The PKI scheme is also supplemented with a particular non-pseudonymous variation for roadside units as well as other adaptations based on Car-2-Car requirements. Finally, the signature verification performance and message sizes are analysed. We argue that the PPKI fulfils the security challenges of Car-2-Car communication and has advantages over a classical PKI.

1 Introduction

Vehicular ad-hoc networks (VANETs), also known as Car-2-Car communication, improve driver safety. Examples of new safety applications enabled by this technology are the timely detection of a traffic jam, collision warnings of obscured vehicles or the warnings about hazards on the road. Once a sufficient saturation of Car-2-Car-enabled vehicles has been reached, road users can automatically warn each other and consequently prevent various accidents. One of the main requirements for a successful rollout of this technology is trustworthiness. Otherwise, if safety messages are not trustworthy, consumers will never adopt this new technology. Beside well-functioning sensors, communication devices and applications, security is a crucial requirement. The main security challenges for the exchange of Car-2-Car messages are authenticity, integrity of messages as well as non-repudiation and privacy.

The following states at first the security requirements for Car-2-Car Communication, the current solution. Afterwards the PPKI Scheme and its adaptations, including a new

authorisation scheme, is explained. Finally, a signature sizes and performance are compared with the current solution.

1.1 Classical Public Key Infrastructures

The present solution, to provide message security, is to implement a classical Public Key Infrastructure (PKI). A certificate authority (*CA*) signs the public keys of the peers, who then digitally sign their messages. A classical PKI solves all security challenges of Car-2-Car communication apart from privacy. To achieve privacy, it is important to prevent the possibility to link messages and peers over a longer time.

1.2 Pseudonymous Public Keys

With a classical PKI, each peer uses its private key to create message signatures. The CA-signed public key will be additionally attached to this signature, which allows other peers to perform a validation. However, the peer's public key represents the main link between all of its messages and, therefore, can be seen as a unique identifier.

The obvious solution is repeatingly to change the public key (and all other identifying attributes). Total anonymity, however, is not desirable, since misbehaving nodes need to be identified and potentially removed from the group of authorized peers (revocation). Such keys are called Pseudonymous Public Keys (PPKs). They fulfil the challenge of non-repudiation.

1.3 Car-2-Car Reference Architecture

The ETSI specifications 102 940 [ETSI1] and 102 941 [ETSI2] define an ITS security reference model, consisting of different types of authorities and peers. The *Enrolment Authority (EA)* is responsible for enabling peers to generally take part in the Car-2-Car communication. The *EA* will be referred to as *CA* within this document. The *Authorisation Authority (AA)* is responsible for authorising a peer "to use of particular application, service, or privilege". For example, the use of specific Car-2-Car safety messages should be subject to those privileges. A *Manufacturer* is an authority that issues each peer a canonical identifier. The *Root CA* issues certificates to all authorities that should take part in the Car-2-Car communication.

1.4 Current Solution

A classical PKI implementation was chosen for the current Pilot PKI of the European Car-2-Car Communication Consortium [ES12]. A vehicle in this infrastructure should receive different types of certificates:

- A long-term certificate, which is only used to authenticate against other authorities.

- A specific amount of short-term certificates which are used as pseudonyms for authentic inter-vehicular communication. Those are issued from so-called *Pseudonym CAs*. Each one is only valid for a given period.
- Authorisation Certificates (Credentials) from the *AA*. They are not available in the current Pilot PKI. Since a credential "should be an unlinkable proof of authentication to the canonical identifier", an obvious solution is, that the peer transfers all of its pseudonymous public keys to the *AA*, which assigns the credentials to each of them.

When all pseudonymous public keys or authorisation credentials have expired, the peer needs to interact again with the *Pseudonym CA* and *AA* to receive another set of keys and credentials.

Peer revocation requires the revocation of the peer's long-term certificate and the prevention of the use of the pseudonymous keys, which can be achieved in two different ways: Firstly, certificate revocation lists (CRLs) can be used, which need to include all the peer's non-expired pseudonymous public keys. The second option is to specify a pseudonymous key update rate so that the peer runs out of certificates very soon.

1.5 Zeng's Pseudonymous PKI (PPKI)

The Pseudonymous PKI for Ubiquitous Computing [Ze06] is a universal scheme for authentic communication with the focus on both effective privacy and traceability in case of misbehaviour. The scheme uses pairing-based cryptography to offer the following features: It enables peer nodes to generate signed pseudonymous public keys (PPKs) themselves, which can only be linked to a peer's identity by the *CA*. The signature of the PPKs can be validated using the global group public key (GPK). Revocation is accomplished by the recalculation of the GPK, which then prohibits the revoked peer to sign new pseudonymous keys. Since different versions of the PPKs cannot validate each other, the peer with the lower version is forced to update its revocation list. The revocation list is then used to calculate the keys according to the new version.

The following explains the cryptographic foundations of pairings, states the PPKI protocols and examines if and how the PPKI can be adapted to Car-to-Car Communication. Also, analysis of the signature sizes and validation performance will be illustrated.

1.6 Pairing-Based Cryptography

Pairing-based cryptography is a rather new cryptosystem. The first pairing-based schemes were developed around the year 2000 [BF00]. The most general form [Ly07] of a pairing is a bilinear map e used for cryptographic applications is the following:

$$e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathcal{G}$$

where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathcal{G}$ are groups of prime order p. It requires the following properties (using $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$) :

- bilinearity, such that $e({g_1}^a, {g_2}^b) = e(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{Z}$
- nondegeneracy, such that $e(g_1, g_2) = 1$ for all g_1 if and only if $g_2 = 1$ or for all for all g_2 if and only if $g_1 = 1$

Using $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, $g \in \mathcal{G}$ and $(a, b, x, y, z) \in \mathbb{Z}_p$, following main cryptographic hard problem-assumptions have been found in bilinear maps [Ly07]:

- Co-Computational Diffie-Hellman (co-CDH) Problem:
  Given $g_1, g_2, {g_2}^x$, compute ${g_1}^x$

- External Diffie-Hellman (XDH) Problem:
  Given $g_1, g_2, g, {g_1}^x, {g_2}^y \; and \; g^z$, decide whether xy = z

- Q-Strong Diffie-Hellman (q-SDH) Problem:
  Given $(g_1, [g_2, {g_2}^x, {g_2}^{2x}, \dots, {g_2}^{qx}])$, find $c, {g_1}^{1/(x+c)}$ for any $c \in \mathbb{Z}_p$

In addition, following theorems are relevant for the PPKI [Ze06]:

- BB Theorem based on the q-SDH Problem:
  Given $g_1, g_2, A = {g_2}^a$, find (t, x) such that $e(t, A \cdot {g_2}^x) = e(g_1, g_2)$

- Zeng's General Theorem based on the BB Theorem:
  Given $g_1, g_2, A = {g_2}^a$ and $(h_1, h_2, \dots, h_k) \in \mathbb{G}_2$, find (t_j, x) such that $\left(t_j, A \cdot {g_2}^x\right) = e(h_j, g_2)$

Bilinear maps are based on elliptic curve cryptography (ECC). The Groups $\mathbb{G}_1$ and $\mathbb{G}_2$ in the pairing e: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathcal{G}$ are subgroups of points on an elliptic curve over a field and $\mathcal{G}$ is a subgroup of a multiplicative group of a finite Field [GPS06].

For the implementation of a pairing, several algorithms exist. The first discovered ones were the Weil and the Tate pairing [Ly07]; however, faster pairing algorithms were developed [LLP08], [Ve10]. Today, the Optimal Ate pairing is one of the most efficent solutions [BG10].

2 The PPKI Scheme

The following scheme was defined in a universal form [Ze06]. It will be explained shortened and with slight modifications where the PPKI scheme was too generic for an implementation. In particular, the PPKI scheme does not require a specific signature algorithm; instead it uses a generic notation for a zero-knowledge proof of knowledge.

The present implementation (see below in section Implementation) uses the Schnorr scheme [Sc89], since Zeng already illustrated its use. In addition, through the utilisation of a self-feedback mode, the scheme makes this implementation very secure [CM09]. However, other signature algorithms can be chosen as well.

CA Initialisation

The *CA* will be initialised using the following procedure: **1.** Generate a random private key $a \in_R \mathbb{Z}_p$. **2.** Compute the public key $A = {g_2}^a \in \mathbb{G}_2$. **3.** Generate the random elements $h_1, h \in_R \mathbb{G}_2$. **4.** Publish the public key $PK_{Ver} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathcal{G}, g_1, g_2, e, h_1, h, A)$ and the current version Ver, which starts at 1.

Peer Initialisation

The peer will be initialised using the following procedure: **1.** Generate a random private-key $x \in_R \mathbb{Z}_p$. **2.** Calculate the root public key $y = h^x$. **3.** Create a peer identifier ID. This should be provided ex-factory and the peer must not be able to change it afterwards. In addition, a proof s_{ID} for the peer's ownership of ID and public key must be generated. This will be discussed later in the Car-2-Car adaptation.

Peer Registration

After the initialisation the peer can register with the CA: **1.** Peer computes $y' = {g_2}^x$ and sends (ID, y, y') to the *CA*. **2.** *CA* verifies that $e(y, g_2) = e(h, y')$ holds and that the proof s_{id} is correct. **3.** *CA* generates a salt $\xi \in_R \mathbb{Z}_p$ and computes $z = Hash(ID|y|s_{id}|\xi)$ (| denotes concatenation). **4.** *CA* computes $\left(t_g = {g_1}^{1/(a+z)}, t_h = (h_1 \cdot h^x)^{1/(a+z)}\right)$. **5.** *CA* stores (ID, y, s_{id}, y', ξ) in a database, to enable the later tracing of PPKs. **6.** *CA* sends certificate $\left(t_g, t_h, z\right)$ to the peer. **7.** Peer verifies the certificate by ensuring that $e\left(t_g, A \cdot {g_2}^z\right) = e(g_1, g_2)$ and $e(t_h, A \cdot {g_2}^z) = e(h_1 \cdot h^x, h_2)$ hold. **8.** Peer computes $v_1 = e(g_1 \cdot h_1, g_2) \in \mathcal{G}$, $v_2 = e\left(t_g \cdot t_h, {g_2}^{-1}\right) \in \mathcal{G}$ and $v_3 = e(h, g_2) \in \mathcal{G}$. Since those terms are required quite often, the storage of their results (v_1, v_2, v_3) brings some performance gain. **9.** Finally, the peer stores the CA-certificate $\left(t_g, t_h, z\right)$ and the accelerators (v_1, v_2, v_3) in accordance with the current version Ver.

Pseudonymous Public Key Generation and Verification

When required, a peer can generate a new pseudonymous public key (PPK) using the following procedure: **1.** Chose a random integer $r \in_R \mathbb{Z}_p$. **2.** Compute the PPK $\left(t = \left(t_g \cdot t_h\right)^r, t_y = t^x\right)$. **3.** Generate the PPK Signature PPK_{sig} using the Schnorr Scheme: Firstly, chose three random Elements $x_1 \in_R \mathbb{Z}_p$, $x_2 \in_R \mathbb{Z}_p$ and $x_3 \in_R \mathbb{Z}_p$. Secondly, compute $R_A = {v_1}^{x_1} \cdot {v_1}^{r \cdot x_2} \cdot {v_3}^{x_3} \in \mathcal{G}$, $R_X = t^{-x_3} \cdot {t_y}^{x_1} \in \mathbb{G}_1$ and the Hash $c_s = Hash(R_A|R_X|Ver) \in \mathbb{Z}_p$. Thirdly, compute the three exponents $s_1 = x_1 - c_s \cdot r \in \mathbb{Z}_p$, $s_2 = x_2 - c_s \cdot z \in \mathbb{Z}_p$ and $s_3 = x_3 - c_s \cdot r \cdot x \in \mathbb{Z}_p$. Those four Elements represent the signature: $PPK_{sig} = (c_s, s_1, s_2, s_3)$. **4.** A verifying peer can validate the PPK_{sig} by computing ${R'}_A = {v_1}^{s_1} \cdot e(t, {g_2}^{-s_2} \cdot A^{c_s}) \cdot {v_3}^{s_3} \in \mathcal{G}$, ${R'}_X = t^{-s_3} \cdot {t_y}^{s_1} \in \mathbb{G}_1$, as well as ${c'}_s = ({R'}_A|{R'}_X|Ver) \in \mathbb{Z}_p$ and by verifying that ${c'}_s = c_s$.

The random integer $r \in_R \mathbb{Z}_p$ allows the generation of p different PPKs. The verification of a PPK_{sig} requires three operations in $\mathcal{G}$, which represents the bottleneck in this protocol, as shown later.

Message Signature Generation and Verification

The previously generated PPK will be used to sign a message. This message has to contain a timestamp, which is provided through the GeoNetworking protocol [ETSI3]. The proving peer has to perform the following actions: **1.** Compute the signature for the message m: Firstly, generate a random element $x_{1m} \in_R \mathbb{Z}_p$. Secondly, compute the message hash c_{sm}: $R_m = t^{x_1} \in \mathbb{G}_1$ $c_{sm} = Hash(R_m, m) \in \mathbb{Z}_p$. Thirdly, compute $s_m = x_{1m} - c_{sm} \cdot x \in \mathbb{Z}_p$. Finally, those two elements form the signature $S_M = (c_{sm}, s_m)$. **2.** A verifying peer can validate the message by computing c'_{sm}: $R'_m = t^{s_m} \cdot t_y{}^{c_{sm}} \in \mathbb{G}_1$, as well as $c'_{sm} = Hash(R'_m, m) \in \mathbb{Z}_p$ and by verifying that $c'_{sm} = c_{sm}$.

Tracing

When a misbehaving peer needs to be identified by linking one of its PPKs to its ID, the CA has to perform the following steps: **1.** Iterate through all database entries, which were created during the peer-registration process and check each y'_i if $e(t, y'_i) = e(t_y, g_2)$ using the $PPK(t, t_y)$. When a match is found, then all corresponding information (ID, y, s_{id}, y', ξ) is also discovered. **2.** The tracing result now can be signed and published by the *CA*. However, it is important to keep the element y' back, since this can be used to find all $PPKs$ of this peer, which would constitute a privacy breach.

Peer Revocation

When a peer was identified, it is possible to ban it from any further Car-2-Car communication. Therefore, the *CA* has to create a new version of its public key: **1.** Recompute the hash $\hat{z} = Hash(\widehat{ID}|\hat{y}|\widehat{s_{id}}|\hat{\xi})$ using the database entry of the identified peer. **2.** Compute its new public key $\tilde{g}_1 = g_1{}^{1/(a+\hat{z})}$ $\tilde{g}_2 = g_2{}^{1/(a+\hat{z})}$ $\tilde{A} = g_2 \cdot \tilde{g}_2{}^{-\hat{z}}$. **3.** Increment Ver and publish the new $PK_{Ver} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathcal{G}, \tilde{g}_1, \tilde{g}_2, e, h_1, h, \tilde{A})$. **4.** Publish the revocation data $(Ver, \tilde{g}_1, \tilde{g}_2, \hat{z})$

Since communication between peers using different versions of the public key is impossible, each peer needs to update its copy of the CA's public key as well as its CA-certificate using the revocation data: **1.** Compute $\tilde{A} = g_2 \cdot \tilde{g}_2{}^{-\hat{z}} \in \mathbb{G}_2$. **2.** Generate the new part of its CA-certificate $t_g = (\tilde{g}_1 / t_g)^{1/(z-\hat{z})} \in \mathbb{G}_1$. **3.** Verify the validity of the revocation data using the same procedure as during the registration process, by checking if $e(t_h, A \cdot g_2{}^z) = e(h_1 \cdot h^x, g_2)$ holds. **4.** Store the new $PK_{ver} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathcal{G}, \tilde{g}_1, \tilde{g}_2, e, h_1, h, \tilde{A})$ and the revocation data. **6.** Update the accelerators $v_1 = e(\tilde{g}_1 \cdot h_1, \tilde{g}_2) \in \mathcal{G}$, $v_2 = e(\tilde{t}_g \cdot t_h, \tilde{g}_2{}^{-1}) \in \mathcal{G}$ and $v_3 = e(h, \tilde{g}_2) \in \mathcal{G}$. **7.** Store the new CA-certificate (t_g, t_h, z) and the updated accelerators (v_1, v_2, v_3) in accordance to Ver. It can be seen that a revoked peer is unable to update the certificate, since $z = \hat{z}$ and consequently a division by zero would occur. Now this peer is unable to take part in the communication as a sender or verifying peer.

3 PPKI Adaptation for Car-2-Car Communication

The following explains the design of a hierarchical PKI architecture using the PPKI, the integration of an AA, which issues pseudonymous authorisation certificates and an efficient non-pseudonymous solution for roadside units (RSUs).

3.1 Hierarchical PKI

As already suggested by other PKI schemes for VANETs [Pa08], the high number of countries in Europe results in a substantial regulatory and administrative effort, which makes it necessary to divide *CAs* into regional authorities. If it will be possible to unite all European countries under one single *CA*, then this concept might be unnecessary. However, regional domains also imply regional and therefore smaller certificate revocation lists.

As previously demanded by the ETSI specification, a *Root CA* is required to establish trust between different regional *CA* and *AA* authorities. The *Root CA* uses a standard ECC-based signature scheme to sign the public keys of the authorities. The creation of signature s for the message m using the signee's private key x and the known generator g will be stated as $s = sign_{g,x}(m)$. An exemplary scheme is the Schnorr Signature [Sc89], which will be used here in the following form: **1.** Generate a random element $k \in_R \mathbb{Z}_p$. **2.** Compute $R_s = g^k \in \mathcal{G}$ and the hash $c_s = Hash(R_s|m) \in \mathbb{Z}_p$. **3.** Compute the element $s = x \cdot c_s + k \in \mathbb{Z}_p$ using private key x. **4.** A verifying peer or authority can validate s by computing $R_s = {g_1}^s \cdot (g^x)^{-c_s} \in \mathcal{G}$, as well as $c'_s = Hash(R_s|m) \in \mathbb{Z}_p$ and by verifying that $c'_s = c_s$.

To reduce the signature length, a hierarchical PKI architecture without chaining is chosen for the design. This requires the *Root CA* to sign the public keys of all involved authorities: $rootCert_{A_i} = sign_{g,r}(A_i)$. Also, the *Root CA* specifies the ECC parameters and the generator g, which are used for all non-pseudonymous signatures (e.g. manufacturer signatures).

To enable peers to register at any regional *CA* it is important to provide a standardised credential that proves them as valid Car-2-Car peers. According to the Car-2-Car reference architecture, a Manufacturer Authority sets the peer's canonical identifier ID. This will be enhanced with the issuance of a proof of ownership of the public key and the ID. The manufacturer creates $s_{ID,i} = \{sign_{g,m}(y_i|ID_i), M = g^m, rootCert_M\}$ for the peer's *ID* and public key. The peer has a private and public key dedicated to the *Root CA*. This set is later used to sign and verif the peer's public keys belonging to a regional CA, which is called the regional public key signature. A requirement for this signatures is, that the ECC parameters and the generator g are valid for the entire PKI. Having this s_{id}, the CA can validate the peer and then issue a certificate for pseudonymous communication within the CA's managed region. A peer only needs to register once at a CA. It then stores the necessary information to easily switch the keys when passing

different administrative regions. In case of a revocation, the s_{ID} needs to be sent to all CAs where the peer will be either locally revoked or excluded from future registrations.

3.2 Authorisation and Credentials

Based on the fundamental work of Verheul [Ve01], the following new pseudonymous credential scheme could be integrated into the PPKI: To retain the anonymity of the peer, the *AA* issues a blind signature for the requester's CA public key. An AA is always assigned to one CA, since it builds upon the CA's public key, to enable efficient pseudonymous certificate proofs by the peer. On the other hand, this means for a peer that changes to another a regional *CA*, that it also has to interact with a corresponding AA to generate new certificates. The credential information $CRED$ contains the Credential String CS, which denotes the specific authorisation and optionally an expiration date, the blinded signature s_{cb}, and the AA's certified public key. A message with one credential now contains the elements:

$$\text{M}=(content, S_M, PPK, PPK_{sig}, CRED)$$

The AA uses its local CA's public key and is set up the following way: **1.** Generate private key $b \in_R \mathbb{Z}_p$ and the public key $B = g_2^b$ **2.** Interact with the $root\ CA$ to get a signature for B: s_B.

A peer can acquire a credential and its signature from an AA: **1.** Peer performs a self-blinding on its CA public key $\check{y} = y^f$, $\check{h}_1 = h_1^f$, $\check{y}' = g_2^{fx}$ and $\check{h}' = g_2^f$ using $f \in_R \mathbb{Z}_p$. **2.** Peer sends to the AA the following: $\check{y}, \check{h}_1, \check{y}', \check{h}'$, its current PPK and PPK_{sig}, as well as the requested credential information CS and a proof of the right of obtainment of the requested credential s_c. The proof s_c can, for instance, be bound to this particular PPK. **3.** AA checks the peer's request, by verifying the PPK using PPK_{sig}, the proof s_c and the blinded public key by checking that $e(t, \check{y}') = e(t_y, \check{h}')$, $e(\check{h}_1, \check{y}') = e(\check{y}, \check{h}')$ and $e(h_1, \check{h}') = e(\check{h}_1, g_2)$ hold. **4.** AA then creates the blind signature using $c_r = hash(CS)$, as well as $\check{s}_c = (\check{h}_1 \cdot \check{y})^{1/(b+c_r)}$ and returns (CS, $\check{s}_c, B, s_B$) to the peer. **5.** Peer obtains the real signature $s_c = {\check{s}_c}^{1/f}$ and stores (CS, s_c, B, s_B).

The PPK and PPK_{sig} generation process will be enhanced with following steps: **1.** Peer performs self-blinding to its certificate signature using the same random integer r as used for the PPK creation: $s_{cb} = {s_c}^r$. The pseudonymous credential is composed of $CRED = (CS, s_{cb}, B, s_B)$. **2.** Peer adds an additional proof of the validity of s_{cb} to the PPK_{sig}: Firstly, calculate $R_c = {v_4}^{x_1} \cdot {v_3}^{x_3} \in \mathcal{G}$ by using the new accelerator $v_4 = (h_1, g_2)$ and use it as additional information for the hash: $c_s = Hash(R_A | R_X | Ver | R_c) \in \mathbb{Z}_p$. The new signature $PPK_{sig} = (c_s, s_1, s_2, s_3)$ has still the same amount of elements, yet is also used as a link between s_{cb}, B and the $c_r = Hash(CS)$. **2.** A verifying peer has to perform the following additional steps during the validation: Firstly, Check if B is already a trusted AA public key and if not, validate B using s_B. Secondly, generate $c_r = hash(CS) \in \mathbb{Z}_p$. Thirdly, compute

$R'_c = {v_4}^{s_1} \cdot {v_3}^{s_3} \cdot e(s_{cb}, A \cdot {g_2}^{c_r}) \in \mathcal{G}$. Finally, calculate the hash using R'_c as an additional concatenated hash input value $c'_s = Hash(R'_A|R'_X|Ver|R'_c) \in \mathbb{Z}_p$. By checking that $c_s = c'_s$, the verifying peer can now trust the PPK and the credential information $CRED$.

The proof was derived the following way:

$$\left({s_{cb}}^{1/r}\right)^{b+c_r} = h_1 h^x$$

$${s_{cb}}^{b+c_r} = (h_1 h^x)^r$$

$$e(s_{cb}, {g_2}^{b+c_r}) = e(h_1 h^x, g_2)^r$$

$$e(s_{cb}, B \cdot {g_2}^{c_r}) = e(h_1, g_2)^r \cdot e(h, g_2)^{rx}$$

The credential signature is not directly traceable, however, since it is bound to the peer's private key, which is proven using the PPK_{sig}, the owner of a credential can always be found out by the CA. The proposed solution of the integration of a credential signature proof into the PPK_{sig} proof is the most efficient when the credentials are always the same. When another credential is attached to a message, the verifying peer has to revalidate the whole PPK_{sig}, since the proof has changed. In cases where different credentials are required, it can be more efficient to separate the proofs. The proof of the PPK_{sig} would be the original one and the proof of the credential would consist of R_X and R_C, so that $c_c = (R_X|R_c) \in \mathbb{Z}_p$. R_X is required to assure that the certificate is bound to the peer's private key x. The proof of this certificate would consist of $(c_c, s_1, s_3) \in \mathbb{Z}_p$. Both solutions also support the concatenation of multiple credentials. For each credential signature a separate R_C has to be computed and concatenated to the hash input. For instance, for n credentials $c_s = (R_A|R_X|Ver|R_{c1}|R_{c2}| \dots |R_{cn}) \in \mathbb{Z}_p$.

3.3 RSU Authentication and Authorisation

The GSIS scheme [LSH07], contains with its RSU authentication and authorisation scheme an efficient non-pseudonymous PKI enhancement. Since RSUs have no privacy requirements, PPKs are not necessary, and a more efficient scheme can be used. Conveniently, the GSIS RSU authentication can be adapted to the PPKI. However, this implementation uses the scheme only to sign the RSU's public key. Like regular peers, RSUs use efficient ECC based signatures for messages. The GSIS scheme uses an identity-based signature, in which the hash of the RSU's ID together with a credential string serves CS as a public key. In this adaption, the credential string and the RSU's public key is also signed by the *CA*. The *CA* not only signs the RSU ID and an optional credential string, but also the RSU's public key.

RSU Initialisation and Registration

For identity-based signatures, it is necessary that the RSU certificate key is created at the CA: **1.** RSU creates its private key $x \in \mathbb{G}_1$ and its public key $y = h^x$ for message signatures. **2.** RSU generates a proof s_{id} for ownership of its identifier ID, and a valid binding of its root public key x, ID and the credential string CS. **3.** *CA* verifies s_{id} and

creates the unique RSU identifier $z = Hash(ID|y|CS) \in \mathbb{Z}_p$ **4.** *CA* creates the RSU's certificate key: $t_r = {g_1}^{1/(a+h_{RSU})}$ using the CA's private key a. **5.** RSU calculates accelerator $v = e(g_1, g_2) \in \mathcal{G}$. **6.** RSU stores x,y,z,t_r and v.

RSU Public Key Signature Generation and Verification

In contrast to normal peers, this signature only needs to be recalculated after a version update. **1.** RSU creates the signature using the following procedure: Firstly, create random element $x_r \in_R \mathbb{Z}_p$. Secondly, compute the hash $c_r = Hash(R_r|Ver) \in \mathbb{Z}_p$ using $R_r = v^{x_r} \in \mathcal{G}$. Finally, compute certificate element $s_r = {t_r}^{x_r+c_r} \in \mathbb{G}_1$. **2.** A verifier can validate the signature $PK_{sig} = (c_r, s_r, y, Ver)$ by computing $z' = Hash(ID|CS|y)$, $R_r' = e(s_r, {g_2}^{z'} \cdot A) \cdot v^{-c_r} \in \mathcal{G}$, $c_r' = Hash(R_r'|Ver)$ and by verifying that $c_r = {c_r}'$.

RSU Message Signature Generation and Verification

RSU message signatures are created using the same procedure as for regular peers using the RSU's private key x the public y and the generator h.

RSU Revocation

It is very important that RSU can be revoked using the same method as used for peer revocation. Otherwise two revocation lists have to be used and the advantage of the update method would be lost. For that reason, the unique identifier z of an RSU is compatible to that of a peer. The *CA* can use the same revocation procedures and publish a new revocation data $(Ver, \tilde{g}_1, \tilde{g}_2, \hat{z})$ to the revocation list.

The RSU uses the same version update protocol as a peer, but instead of $\widetilde{t_g}$ it calculates its new private key $\widetilde{t_r} = (\tilde{v}/t_r)^{1/(z-\hat{z})}$ with the new accelerator $\tilde{v} = e(\tilde{g}_1, \tilde{g}_2)$. It can be seen again, that a revoked RSU cannot perform the calculation of $\widetilde{t_r}$.

4 Analysis

The two central questions for the utilisability of the PPKI are: how much the signature sizes of the PPKI differ from a classical PKI scheme and whether the PPKI has a sufficient performance. Both depend on the selected key lenghts. As explained above, the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of an elliptic curve over finite fields, the third group $\mathcal{G}$, however, is a subgroup of a finite field. Hence, two cryptographic key lengths are relevant: Firstly, for the size of p for elliptic curve based cryptography and secondly, the size of the field p^k for the multiplicative subgroup $\mathcal{G}$. To provide security until the year 2030, key lengths for p of 224 bit and for p^k of 2432 bit are recommended, according to ECRYPT II [Sm11]. This requires the implementation of Barreto-Naehrig elliptic curves [BN06], with the parameter $k = 12$, which results in a length of 2688 bit for p^k.

4.1 Signature and Credential Size

A peer's message signature package consists of the message signature itself, the PPK, the PPK signature and the version identifier. Table 1 shows how the signature size is determined. Due to *point compression* for elliptic curves, elements of $\mathbb{G}_1$ can be reduced to

the x-value plus an indicator bit for the y-value. An ECC-based solution would consist of two elements for the message signature, one element for the peer's public key and two elements for the public key signature. This size would be $5 \cdot log_2(p)$, which is 1120 bit using a key size of 224 bit. The PPKI's signature is 66 % larger than an ECC-based one. The non-pseudonymous RSU signature, as shown in Table 2, has about the same size as a regular ECC-based one. The difference depends on the size of the version identifier and the credential string. The *RSU-ID* is already part of the Car-2-Car protocol. Table 3 shows the determination of the authorisation certificate. It is easy to see that the *AA* public key $B \in \mathbb{G}_1$ is very large compared to the other elements, although due to *six-to-one point compression* of Baretto-Naehrig curves [BN06] it is only double the size instead of twelve times. The other sizes are comparable to an ECC-based solution. Since the public keys of authorities do not change very often, a possible solution to avoid this problem is to distribute public keys separately. The credential then would only need to refer to the required key using a small identifier.

Table 1: Components and sizes of the PPKI peer signature

Signature Parts	Elements	Relative Size	Example for $p \triangleq 224$ and $ver \triangleq 32$ (bit)
Message Signature	$c_{sm} \in \mathbb{Z}_p$ $s_{sm} \in \mathbb{Z}_p$	$\log_2(p)$ $\log_2(p)$	224 224
Pseudonymous Public Key	$t \in \mathbb{G}_1$ $t_y \in \mathbb{G}_1$	$\log_2(p) + 1$ $\log_2(p) + 1$	225 225
Pseudonymous Public Key Signature	$c_s \in \mathbb{Z}_p$ $s_1 \in \mathbb{Z}_p$ $s_2 \in \mathbb{Z}_p$ $s_3 \in \mathbb{Z}_p$	$\log_2(p)$ $\log_2(p)$ $\log_2(p)$ $\log_2(p)$	224 224 224 224
Version	ver	$\log_2(ver)$	32
		$8\log_2(p) + \log_2(ver) + 2$	1826

Table 2: Components and sizes of the PPKI RSU signature

Signature Parts	Elements	Relative Size	Example for $p \triangleq 224$, $ver \triangleq 32$ and CS $\triangleq 64$ (bit)
Message Signature	$c_{sm} \in \mathbb{Z}_p$ $s_{sm} \in \mathbb{Z}_p$	$\log_2(p)$ $\log_2(p)$	224 224
Public Key	$y \in \mathbb{G}_1$	$\log_2(p) + 1$	225
Public Key Signature	$c_r \in \mathbb{Z}_p$ $s_r \in \mathbb{Z}_p$	$\log_2(p)$ $\log_2(p)$	224 224
Version Cred. String	ver CS	$\log_2(ver)$ $log_2(CS)$	32 64
		$6\log_2(p) + \log_2(ver + CS) + 1$	1217

Table 3: Components and sizes of the PPKI credential

Signature Parts	Elements	Relative Size	Example for $p \triangleq 224$ and CS $\triangleq$ 64 (bit)
Cred. String and Signature	CS	$\log_2(CS)$	64
	$s_{cb} \in \mathbb{G}_1$	$\log_2(p) + 1$	225
AA Public Key with *Root CA*'s Signature	$B \in \mathbb{G}_2$	$\log_2(p^2) + 1$	449
	$c_{rb} \in \mathbb{Z}_p$	$\log_2(p)$	224
	$s_{rb} \in \mathbb{Z}_p$	$\log_2(p)$	224
		$\log_2(CS) + 5\log_2(p) + 2$	1186

4.2 Performance

The main performance constraints in this protocol are the operations in $\mathcal{G}$ and the pairing e. Since signing and validating are the operations most often performed on the peer side, they will be subject to this benchmark. Compared to the validation process, the signature process misses one pairing operation. This means that the validation process is computationally more expensive than the signature creation. Hence, the sign operation can be neglected, especially since it is only rarely used. Message signature operations are performed in $\mathbb{G}_1$ and are equal to a common ECC-based Schnorr implementation. The time required for a message signature validation serves as a reference for the PPK signature performance. The same applies to the authorisation credential verification. Since no other suitable pseudonymous authorisation scheme was found, the only comparable solution is short-term ECC-based certificates, similar to the short-term public keys. Again, the message verification time is taken as a reference. The benchmark was performed on a 2.7 GHz Intel i7 CPU As Table 4 shows, the validation times and ratios using a 254-bit on a Barreto-Naehrig curve using a performant Optimal Ate pairing library [BG10]. Since automotive implementations will not use PC CPUs but cost and energy efficient processors or application-specific integrated circuits (ASICs), the ratios of the different verification processes are more important than the measured times.

Table 4: Performance of PPKI Operations on a Barreto-Naehrig curve with $p \triangleq 224$ bit

Operation	Time / ms	ratio
Message Verification	0.426	1
PPK Verification	3.956	9.27
RSU PK Verification	2.052	4.82
Cred. Verification	3.004	7.05

The result shows that the PPK signature verification is about *ten times* slower than the message verification process. The slowdown usually affects only the first message of a new peer, because only at this point a PPK verification is required. Subsequently, until the next PPK change, only message signatures need to be validated. Nevertheless, this is a situation requiring further investigation: since a PPK change is only reasonable when all peers within a certain range perform it at the same time, the performance can become critical. It may help to specify the maximum size of the group of peers which execute the PPK change. The RSU public key verification is *twice as fast* as the PPK verification.

A pseudonymous authorisation credential verification is *seven times* slower than short term ECC based certificates, however still faster than a PPK verification. Safety relevant messages that contain an authorisation credential require both to be verified: the PPK and the authorisation credential. Together it would take 6.96 ms for the verification, which is *eight times* more than two ECC-based verifications.

5 Conclusion

In this work, it could be shown that the PPKI does indeed meet the challenges of Car-2-Car communication. Message authenticity, integrity, and non-repudiation are fulfilled equally. It is possible to set up a hierarchical PKI architecture with regional CAs and multiple *AAs*.

The new developed pseudonymous authorisation scheme allows to integrate trusted authorisation authorities into the PPKI. They enable peers to use special safety messages or commercial services like tolling, information or entertainment services. Similarly to the PPKI authorisation it allows peer side pseudonymous credential generation with about the same performance requirements. Without such a solution, the amount of pseudonymous authentication credentials for multiple services can become enormous, and the issuing process would be computational expensive, since each pseudonymous public key requires a unique authentication credential. On-demand issuing services [AGL13] may be practically applicable for some commercial services, however they would be much more complex then the PPKI solution. An important privacy feature of the new pseudonymous authorisation scheme is the optionally “blinded” registration process that prevents the authorisation authorities to link the issued credentials to a peer. If a customer pays a tolling authority for a one-year certificate, the authority does not need to know the peer's identity. The privacy of this process can be compared to the process of buying a tolling vignette.

In contrast to the classical PKI, the PPKI allows the peers to change their pseudonym at any required frequency. In addition, classical PKI solutions require an Internet connection to distribute pseudonyms and authorisation certificates and maybe also to provide CRLs. After the initial registration at a CA and AA, a PPKI peer can rely only on Car-2-Car communication. This makes the solution robust and user-friendly. Also, the revocation process is straightforward: peers are forced to distribute and import revocation data, otherwise they can no longer take part in the communication.

Nevertheless, those features come with additional computational costs for key and credential validation. A PPKI hardware solution will probably be more expensive, especially since a pairing implementation is more complicated than a plain ECC based one. A quantitative determination of costs and benefits cannot be made since costs could be reduced by mass production of ASICs. Although the system in the car becomes more sophisticated, it carries out tasks that would otherwise be located on the infrastructure side. From a qualitative point of view, the PPKI scales better, the communication overhead is reduced and administrative processes become easier to manage.

References

[AGL13] Alexiou, N.; Gisdakis, S.; Laganà, M.; Papadimitratos, P.: Towards a secure and privacy-preserving multi-service vehicular architecture: WOWMOM 2013, IEEE

[BN06] Barreto P.; Naehrig M.: Pairing-Friendly Elliptic Curves of Prime Order: Selected Areas in Cryptography - SAC 2005; Lecture Notes in Computer Science: Springer, 2006

[BF00] Boneh D.; Franklin M.: Identity-Based Encryption from the Weil Pairing: Advances in Cryptology, Springer Berlin Heidelberg, 2001

[BG10] Beuchat, J.; Gonzáles-días, J.; Mitsunari, S.; Okamoto, E.; Rodríguez-henríquez F.; Teruya, T.: High-speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves: Proceedings of the 4th International Conference on Pairing-based Cryptography, Pairing'10, Springer-Verlag Berlin, Heidelberg 2010

[CM09] Cao Z.; Markowitch O.: Security Difference between DSA and Schnorr's Signature: Networks Security, Wireless Communications and Trusted Computing: IEEE, 2009

[ES12] Escrypt: Efficient Public Key Infrastructure (PKI) Solutions for Embedded Systems: Website Report: https://www.escrypt.com/company/single-news/detail/efficient-public-key-infrastructure-pki-solutions-for-embedded-systems/

[ETSI1] ETSI Specification 102 940: Intelligent Transport Systems; Security; ITS communications security architecture and security management: European Telecommunications Standards Institute, 2012

[ETSI2] ETSI Specification 102 941: Intelligent Transport Systems; Security; Trust and Privacy Management: European Telecommunications Standards Institute, 2013

[ETSI3] ETSI Specification 102 636-4-1: Intelligent Transport Systems; Vehicular communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1 Media-Independent Functionality: European Telecommunications Standards Institute, 2011

[GPS06] Galbraith S.; Paterson K.; Smart N.: Pairings for Cryptographers: Cryptology ePrint Archive, Report 2006/165, 2006

[JPBC] Angelo De Caro: The Java Pairing-Based Cryptography Library: Università degli Studi di Salerno

[LLP08] Lee E.; Lee H.; Park C.: Efficient and generalized pairing computation on abelian varieties: Cryptology ePrint Archive, Report 2008/040, 2008.

[LSH07] Lin, X.; Sun, X.; Ho, P.; Shen, X.; GSIS: A Secure and Privacy-Preserving Protocol for Vehicular Communications: IEEE Transactions on Vehicular Technology, 56, 2007

[Ly07] Lynn, B: On the implementation of pairing-based cryptosystems: Stanford University, 2007

[Sm11] Smart, N.: ECRYPT II Yearly Report on Algorithms and Keysizes: European Network of Excellence in Cryptology II, 2011

[Pa08] Papadimitratos, P. et al: Secure vehicular communication systems: design and architecture: IEEE Communications Magazine, Vol 46, Issue 11: IEEE Press, Piscataway: 2008, pp 100-109

[Sc89] Schnorr C.: Efficient Identification and Signatures for Smart Cards, Advances in Cryptology, Springer Berlin Heidelberg, 1989

[Ve01] Verheul, E.: Self-Blindable Credential Certificates from the Weil Pairing: ASIACRYPT '01, Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, Springer-Verlag London 2001; pp. 533-551

[Ve10] Vercauteren F.: Optimal Pairings: IEEE Transactions on Information Theory: 2010; pp 455-461

[Ze06] Zeng, K.: Pseudonymous PKI for Ubiquitous Computing: EuroPKI 2006, LNCS 4043. Springer-Verlag Berlin Heidelberg 2006; pp. 207-222