The Unified Butterfly Effect

Efficient Security Credential Management System for Vehicular Communications

Marcos A. Simplicio Jr.[1], Eduardo Lopes Cominetti[1], Harsh Kupwade Patil[2], Jefferson E. Ricardini[1] and Marcos Vinicius M. Silva[1]

[1] Escola Politécnica, Universidade de São Paulo, Brazil.
{mjunior,ecominetti,joliveira,mvsilva}@larc.usp.br
[2] LG Electronics, USA. harsh.patil@lge.com

Abstract. Security and privacy are important requirements for the broad deployment of intelligent transportation systems (ITS). This motivated the development of many proposals aimed at creating a Vehicular Public Key Infrastructure (VPKI) for addressing such needs. Among them, schemes based on pseudonym certificates are considered particularly prominent: they provide data authentication in a privacy-preserving manner while allowing vehicles to be revoked in case of misbehavior. Indeed, this is the approach followed by the Security Credential Management System (SCMS), one of the leading candidate designs for protecting vehicular communications in the United States. Despite SCMS's appealing design, in this article we show that it still can be further improved. Namely, one of the main benefits of SCMS is its so-called butterfly key expansion process, which allows batches of pseudonym certificates to be issued for authorized vehicles by means of a single request. Whereas this procedure requires the vehicle to provide two separate public/private key pairs for registration authorities, we present a modified protocol that uses a single key for the same purpose. As a result, the processing and bandwidth costs of the certificate provisioning protocol drop as far as 50%. Such performance gains come with no negative impact in terms of security, flexibility or scalability when compared to the original SCMS.

Keywords: Vehicular communications · security and privacy · pseudonym certificates · butterfly key expansion · Security Credential Management System

1 Introduction

In the last decade, the automotive industry has been improving the computation and communication capabilities of embedded in vehicles (e.g., sensors and actuators) and roadside units (e.g., cameras, radars, and dynamic displays). In particular, the growing support for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications, collectively called V2X technologies, allows the development of many applications targeted at intelligent transportation systems (ITS) [IKRK08, HPY+14]. Such applications usually involve the metering of road conditions (e.g., its slope and current traffic) as well as the vehicles' state (e.g., velocity, acceleration, position and distance to other cars) [PFE+09]. The information acquired can then be shared among vehicles and further relayed to drivers, facilitating timely intervention – either manually or (semi-)automatically – for enhancing transportation safety and efficiency. Enabling such applications is among the goals of standards such as the Wireless Access in a Vehicular Environment (WAVE), designed to be used with the Dedicated Short Range Communications (DSRC) protocol [JD08].

Albeit promising, the large scale deployment of ITS technologies still faces some important challenges, especially security and privacy concerns [SMK09, FKL14]. For example, the authenticity of data exchanged via V2X is critical to prevent abuse of the system by malicious users (e.g., forcing a vehicle to stop by simulating an accident ahead). This can be accomplished by means of digital signatures on broadcast messages, as well as revocation of misbehaving vehicles' certificates whenever they are detected. In this case, vehicles would only trust in, and act upon, messages signed by non-revoked peers. If traditional, long-term certificates are employed for this purpose, however, the users' privacy can be put at risk. For example, since many ITS applications require vehicles to periodically broadcast their positions, their mobility patterns can be tracked by eavesdroppers. Actually, even if exact positions are not shared via V2X, vehicle tracking is still possible with some accuracy if the eavesdropper pinpoints where and when messages signed by the target were broadcasted.

Aiming to cope with such authentication and privacy requirements, many proposals appeared in the literature for creating a Vehicular Public Key Infrastructure (VPKI) (for a survey, see [KP15]). Among them, one approach of particular interest relies on pseudonym certificates [PSFK15]. Differently from traditional certificates, pseudonym ones do not contain any information that could be easily associated with their owners. Therefore, such credentials can be used for signing messages broadcast for other vehicles without compromising their owner's privacy. The usage of long-term credentials is then reserved for situations in which a vehicles must be identified, such as proving that it is authorized to obtain new pseudonym certificates.

Among the many pseudonym-based security solutions for V2X (see [PSFK15] for a survey), one of the most prominent is the Security Credential Management System (SCMS) [WWKH13, CAM16]. Actually, this solution is today one of the leading candidate designs for protecting vehicular communications in the United States [CAM16]. In SCMS, a Registration Authority (RA) creates batches of pseudonym certificates for authorized vehicles from a single request, in the so-called butterfly key expansion process. The RA shuffles those certificates together and sends them to a Pseudonym Certificate Authority (PCA). The certificates are then individually signed and encrypted by the PCA before being delivered to the requesting vehicle. The system is designed in such a manner that, unless RA and PCA collude, they are unable to link a pseudonym certificate to its owner, nor learn whether or not two pseudonym certificates belong to the same vehicle. Specifically, the PCA does not identify the owner of each certificate, whereas the RA that delivers the certificate to the vehicle does not learn its inner contents. In case of abuse, however, the privacy of the misbehaving vehicle is lifted and its certificates are revoked in an efficient manner, by placing a small piece of information in a Certificate Revocation List (CRL).

Even though SCMS provides an efficient and scalable security framework for vehicular communications, in this article we show that its design can be further improved. Namely, we describe a method by means of which the processing and bandwidth costs of SCMS's certificate provisioning protocol can be reduced approximately by half. Basically, this gain is obtained when, instead of using separate butterfly keys for encryption and signature, both keys are combined in a unified key derivation process. Besides describing the solution in detail, we show that the performance improvements come with no negative impact in terms of security, flexibility or scalability when compared to the original SCMS protocol.

The remainder of this article is organized as follows. Section 2 summarizes the notation employed along the document. Section 3 discusses related works, giving a broad view of the state-of-the-art on V2X security. Section 4 describes with some detail the SCMS protocol, in particular its butterfly key expansion process. Section 5 then presents our proposed improvements to the original SCMS, analyzing the resulting security and performance. Finally, section 6 concludes the discussion and presents ideas for future work.

2 General Notation

For convenience of the reader, Table 1 lists the main symbols and notation that appear along this document.

Whenever pertinent, we assume standard algorithms for data encryption hashing and digital signatures. In particular, we assume that the following algorithms are employed: symmetric encryption (i.e., using shared keys) is done with the AES block cipher [NIS01] whereas asymmetric encryption (i.e., involving public/private key pairs) is performed with ECIES [IEE04]; hashing is performed with SHA-2 [NIS15a] or SHA-3 [NIS15b]; and the creation and verification of digital signatures uses algorithms such as ECDSA [NIS13] or EdDSA [BDL+12, JL17].

Table 1: General notation and symbols

Symbol	Meaning
G	The generator of an elliptic curve group
sig	A digital signature
$cert$	A digital certificate
$\mathrm{U}, \mathcal{U}$	Public signature keys (stylized $\mathcal{U}$: reserved for PCA)
u, u	Private keys corresponding to U and $\mathcal{U}$ (respectively)
S	Public caterpillar key for signature
s	Private caterpillar key for signature
E	Public caterpillar key for encryption
e	Private caterpillar key for encryption
$\hat{S}$	Public cocoon key for signature (derived from caterpillar key)
$\hat{s}$	Private cocoon key for signature (derived from caterpillar key)
$\hat{E}$	Public cocoon key for encryption (derived from caterpillar key)
$\hat{e}$	Private cocoon key for encryption (derived from caterpillar key)
β	Number of cocoon keys in pseudonym certificate batch
la_id	ID of a Linkage Authority (LA)
$\mathtt{ls}_i$	Linkage seed
$\mathtt{plv}_i$	Pre-linkage value
$\mathtt{lv}$	Linkage value
τ	Number of time periods covered by pseudonym certificate batch
σ	Number of certificates valid at any time period
$Enc(\mathcal{K}, str)$	Encryption of bitstring str with key $\mathcal{K}$
$Dec(\mathcal{K}, str)$	Decryption of bitstring str with key $\mathcal{K}$
$Sign(\mathcal{K}, str)$	Signature of bitstring str, using key $\mathcal{K}$
$Ver(\mathcal{K}, str)$	Verification of signature on str, using key $\mathcal{K}$
$Hash(str)$	Hash of bitstring str
$str_1 \parallel str_2$	Concatenation of bitstrings str_1 and str_2

3 Related Works

The emergence of V2X has led to the development of many proposals addressing security and privacy issues in this scenario. The most promising ones are based on pseudonyms, using strategies such as symmetric, asymmetric and identity-based cryptography [Sha85, ZAFB12], as well as group signatures [CvH91] (for a comprehensive survey, see [PSFK15]). To give an overview of the state-of-the-art, in what follows we discuss some recent works in the area.

The main goal of the pseudonym scheme with user-controlled anonymity (PUCA) [FKL14] is to ensure the end users' privacy even toward (colluding) system entities. In PUCA, the revocation procedure assumes that each vehicle's trusted module, responsible for managing its long-term enrollment certificate and other cryptographic keys, suspends its operation after receiving an order of self-revocation (OSR) addressed to one of its pseudonyms. As a result, no pseudonym resolution is necessary and, thus, the identity of the users is preserved even in case of misbehavior. Albeit interesting, it is unclear how the system would prevent attackers from filtering out OSR messages addressed to them, thus avoiding revocation. In addition, by design the authors prevent pseudonym certificates from being linked together, arguing that traditional investigation methods should be used in case of misbehavior rather than rely on the V2X system to identify the culprit. This reasoning would be adequate if V2X itself did not introduce new threats and misbehavior capabilities, but that is not the case: after all, malicious users can abuse the system to cause collisions or facilitate robbery (e.g., by inducing other vehicles to slow down or take an alternate route, claiming an accident nearby). Therefore, it is reasonable that the system itself provides mechanisms to solve the issues it creates, which motivates the need of revocable privacy.

Another interesting example is the Issue First Activate Later (IFAL) scheme [Ver16], which proposes that pseudonym certificates need to be activated before they can be used by the vehicles. In this case, only honest devices would periodically receive activation codes. This avoids the growth in size (or even the need) of CRLs, since even if a vehicle receives a large batch of pseudonym certificates valid for a long time, it still needs to obtain the corresponding activation codes to use those certificates. As a result, a revoked vehicle's certificates that have not yet been activated do not need to be included in a CRL, whereas the information identifying certificates that are already active only need to appear in a CRL until they expire. One limitation of this approach is that it obliges vehicles to periodically contact the V2X infrastructure. However, since activation codes can be very small, this process should be much less cumbersome than the periodical delivery of small batches of certificates, a possible alternative to avoid issuing certificates to revoked devices. This promising characteristic of IFAL is, however, counterbalanced by a security issue: the certificate authority that issues pseudonym certificates, even without colluding with any other entity, can link the pseudonym certificates it issues to the corresponding device's enrollment certificates; therefore, the users' privacy depends on that authority's willingness to delete this information.

Like IFAL, the Binary Hash Tree based Certificate Access Management (BCAM) scheme [KPW17] was also designed to reduce CRL sizes by means of activation codes. Unlike IFAL, however, BCAM was designed to interoperate with the SCMS architecture, inheriting its ability to protect the privacy of honest users against any non-colluding system entities. In addition, BCAM allows activation codes to be recovered from a small piece of information broadcast by a Certificate Access Manager (CAM), so vehicles are not required to explicitly request them. Specifically, each batch of certificates issued to a given vehicle is encrypted by CAM, and the decryption key can be computed from a device specific value (DSV) generated by the CAM using a binary tree. By broadcasting the tree's root, all vehicles can decrypt their own batches. To revoke a misbehaving vehicle, the nodes of the tree that would allow the corresponding DSVs to be computed are not broadcast, thus preventing the decryption of that vehicle's certificates. This efficient revocation process provided by BCAM applies not only to SCMS's original butterfly key expansion process, but also to the unified approach hereby proposed. Hence, BCAM can be seen as complementary to our solution.

The Security Credential Management System (SCMS), originally proposed in [WWKH13] and later extended in [CAM16], is one of the few schemes in the literature that deals with revocable privacy while preventing any given entity from tracking devices all by itself

(i.e., without colluding with other system entities). By doing so, it copes with security needs of V2X while elegantly addressing a threat model in which the system's entities can be considered "honest-but-curious": they follow the correct protocols but may engage in passive attacks, such as trying to infer sensitive information from the exchanged data aiming to track vehicles [KP15]. Since this is one of the leading candidate designs for protecting V2X security in the US [WWKH13, CAM16], and is used as the basis for our own work, we analyze SCMS more closely in the following section.

4 The Security Credential Management System

In this section, we describe SCMS in more detail. Along the discussion, we focus on the description given in [WWKH13] rather than in [CAM16]. The motivations for this approach are that (1) the former displays a more concise notation and (2) the modifications introduced in the latter do not affect our high-level description, nor the improvements hereby described. Nevertheless, for completeness, whenever pertinent we also briefly mention eventual modifications of [WWKH13] done in [CAM16].

In SCMS, each device receives two types of certificates: an enrollment certificate, which have long expiration times and identify valid devices in the system; and multiple pseudonym certificates, each having a short validity (e.g., a few days), in such a manner that $\sigma \geqslant 1$ pseudonym certificates may be valid simultaneously. For protecting its privacy, a particular vehicle should frequently change the pseudonym certificate employed in its communications, thus avoiding tracking by nearby vehicles or by roadside units. In practice, the system should limit the value of σ to a small number to avoid "sybil-like" attacks [Dou02], in which one vehicle poses as a platoon aiming to get some advantage over their peers [MLHS12, AGMM15]. For example, such a fake platoon could end up receiving preferential treatment from traffic lights programmed to give higher priority to congested roads.

The solution was designed to allow the distribution of multiple pseudonym certificates to vehicles in an efficient manner, while providing mechanisms for easily revoking them in case of misbehavior by their owners. For this purpose, SCMS relies basically on the following entities (see Figure 1 for a complete architecture and [WWKH13] for the description of all of its elements):

- Pseudonym Certificate Authority (PCA): responsible for issuing pseudonym certificates to devices.
- Registration Authority (RA): receives and validates requests for batches of pseudonym certificates from devices, identified by their enrollment certificates. Those requests are individually forwarded to the PCA, in such a manner that requests associated to different devices are shuffled together so the PCA cannot link a set of requests to a same device.
- Linkage Authority (LA): generates random-like bitstrings that are added to certificates so they can be efficiently revoked (namely, multiple certificates belonging to a same device can be linked together by adding a small amount of information to certificate revocation lists – CRLs). SCMS uses two LAs, even though its architecture supports additional LAs.
- Misbehavior Authority (MA): identifies misbehavior patterns by devices and, whenever necessary, revokes them by placing their certificate identifiers into a CRL.

These entities play different roles in the two main procedures provided by SCMS: the butterfly key expansion, which allows pseudonym certificates to be issued; and key linkage, which allows the efficient revocation of malicious vehicles. Both are described in the following subsections.

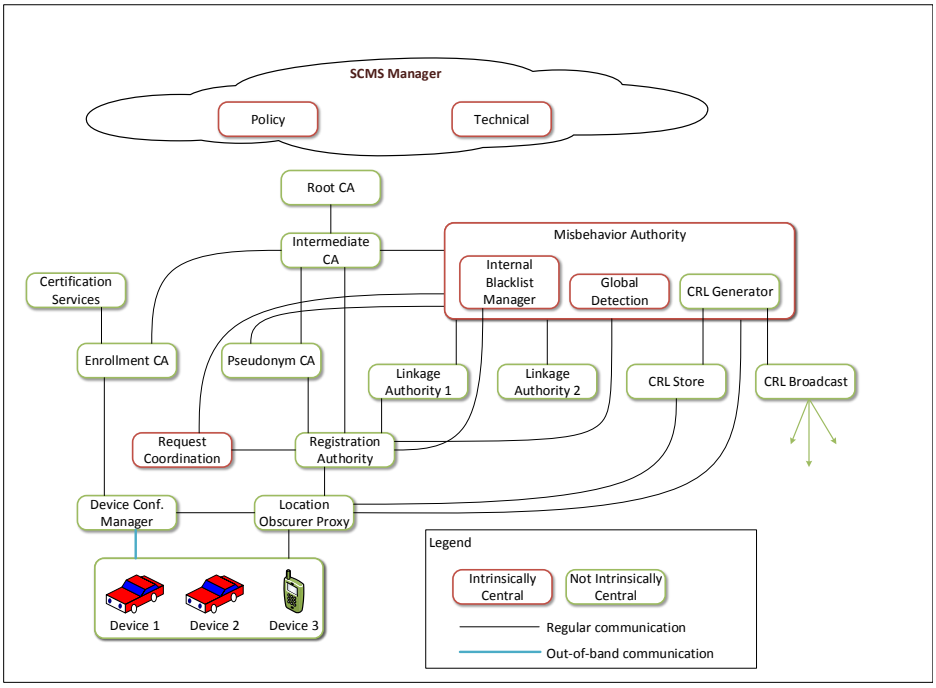

Figure 1: SCMS overview. Source:[WWKH13].

4.1 Butterfly key expansion

The pseudonym certification provisioning process in SCMS provides an efficient mechanism for devices to obtain arbitrarily large batches of (short-lived) certificates with a small-sized request message. It comprises the following steps, as illustrated in Figure 2. First, the device generates two *caterpillar* private/public key pairs, $(s, S = s \cdot G)$ and $(e, E = e \cdot G)$. The public caterpillar keys S and E are then sent to the Registration Authority (RA) together with two suitable pseudorandom functions (PRF) f_s and f_e. The key S is employed by the RA in the generation of β public *cocoon* signature keys $\hat{S}_i = S + f_s(i) \cdot G$, where $0 \leqslant i < \beta$ for an arbitrary value of β; similarly, the RA uses E for generating β public cocoon encryption keys $\hat{E}_i = E + f_e(i) \cdot G$. The exact manner by which f_s and f_e are instantiated in [WWKH13] differs from the description given in [CAM16], but since the differences are irrelevant for our discussion, we hereby omit their internal details. Pairs of cocoon keys $(\hat{S}_i, \hat{E}_i)$ from different devices are then shuffled together by the RA and sent individually to the Pseudonym Certificate Authority (PCA) for the generation of the corresponding pseudonym certificates.

After receiving a pair of cocoon keys from the RA, the PCA can either create explicit certificates or engage in a implicit certification process [Cer13]. For explicit certificates, the PCA computes the vehicle's public signature key as $\mathrm{U}_i = \hat{S}_i + r_i \cdot G$, for a random value r_i, inserts U_i into a certificate $cert_i$ containing the required metadata `meta` (e.g., the certificate's validity period and linkage values, discussed later in Section 4.2), and digitally signs $cert_i$ with its own private key u. The $\hat{E}_i$ key is then used to encrypt the signed certificate together with the value of r_i; hence, only the requesting vehicle can decrypt the result to learn U_i and compute the corresponding private signature key $\mathrm{u}_i = s + r_i + f_s(i)$.

For implicitly certified keys, this process is slightly different: the PCA starts by computing a credential $V_i = \hat{S}_i + r_i \cdot G$ again for a random r_i, and then creates the implicit certificate $cert_i = (V_i, \texttt{meta})$; the PCA then signs this certificate to obtain $sig_i = h_i \cdot r_i + u$, where $h_i = Hash(cert_i)$, and sends back to the vehicle the pair $(cert_i, sig_i)$ encrypted with

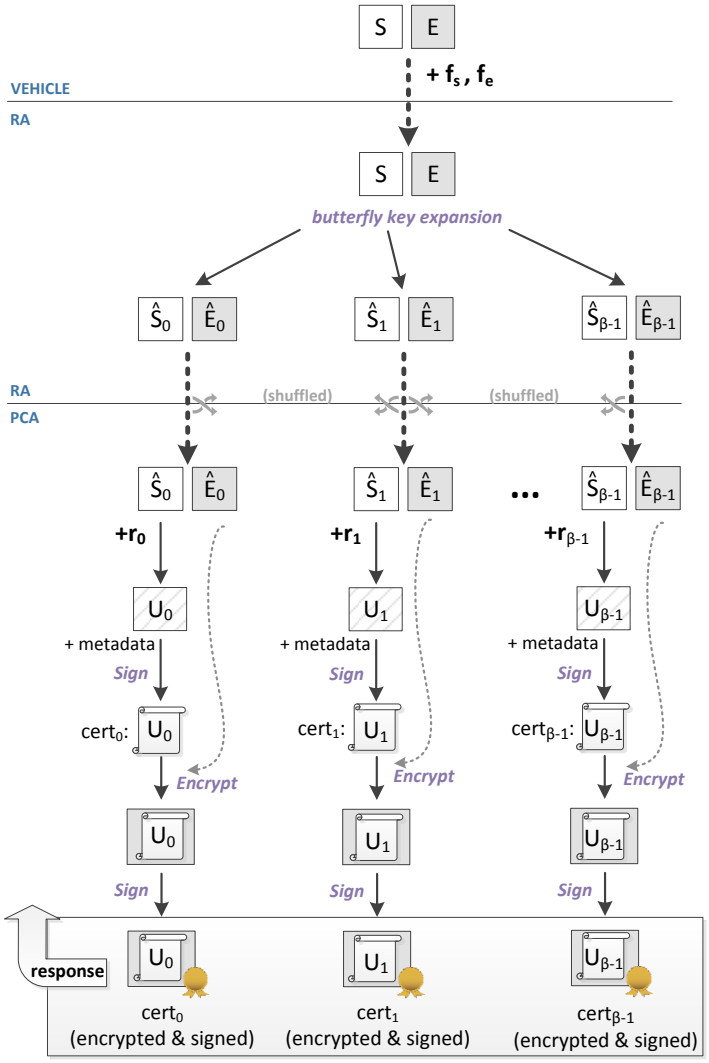

Figure 2: SCMS's butterfly key expansion and certificate generation.

$\hat{E}_i$. The vehicle, after decrypting the PCA's response, computes $h_i = Hash(cert_i)$ and sets its own private signature key as $u_i = h_i \cdot (s + f_s(i)) + sig_i$, whereas the corresponding public signature key takes the form $U_i = u_i \cdot G$. The validity of the public key U_i can then be implicitly verified by ascertaining that $U_i = h_i \cdot V_i + \mathcal{U}$, where $\mathcal{U}$ is the PCA's public signature key.

Whichever the certificate model adopted, the encrypted PCA's response is also signed using its own private signature key, aiming to prevent an "honest-but-curious" RA from engaging in a Man-in-the-Middle (MitM) attack. Namely, without this signature, a MitM attack by the RA could be performed as follows: (1) instead of $\hat{E}_i$, the RA sends to the PCA a fake cocoon encryption key $\hat{E}_i^* = z \cdot G$, for an arbitrary value of z; (2) the RA decrypts the PCA's response using z, learning the value of $cert_i$; (3) the RA re-encrypts the certificate with the correct $\hat{E}_i$, sending the result to the vehicle, which proceeds with the protocol as usual; and (4) whenever a vehicle presents a pseudonym-based $cert_i$ to its counterparts, so they can validate its own public signature key U_i, the RA can link that $cert_i$ to the original request, thus identifying the corresponding vehicle. As long as the vehicle verifies the PCA's signature on the received response, however, such MitM

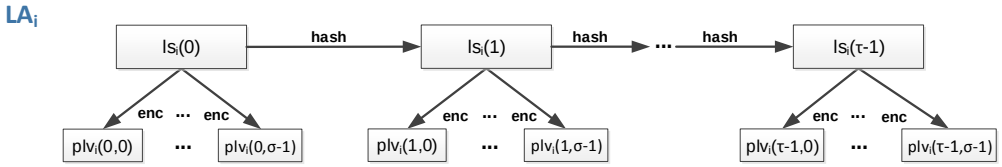

Figure 3: SCMS's key linkage process: generation, by LA_i, of pre-linkage values employed for certificate revocation.

attempt would fail because the RA would not be able to provide a valid signature for the re-encrypted certificate generated in the attack's step 3.

Also independently of the type of certificate adopted, the users' privacy is protected in this process as long as the RA and PCA do not collude. After all, the shuffling of public cocoon keys performed by the RA prevents the PCA from learning whether or not a group of keys in the batch belong to a same device. Unlinkability of public keys towards the RA, in turn, is obtained because the latter does not learn the value of $cert_i$ in the PCA's encrypted response.

4.2 Key linkage

To avoid large certificate revocation lists (CRLs), revocation is done in such a manner that many certificates from a same user can be linked together by inserting only a small amount of information into a CRL. For this purpose, each certificate receives a linkage value $\texttt{lv}$, computed by XORing ℓ pre-linkage values $\texttt{plv}_i$ (where $1 \leqslant i \leqslant \ell$) provided by $\ell \geqslant 2$ different Linkage Authorities (LA). The generation of $\texttt{plv}_i$ by LA_i is done upon request by the RA, as follows.

First, as illustrated in Figure 3, LA_i picks a random, 128-bit linkage seed $\texttt{ls}_i(0)$. Then, if the RA's request covers τ certificate time periods, LA_i iteratively computes a τ-long hash chain [Lam81] $\texttt{ls}_i(t) = \mathit{Hash}(la\_id_i \,\|\, \texttt{ls}_i(t-1))$, where la_id_i is LA_i's identity string and $1 \leqslant t < \tau$. Each $\texttt{ls}_i(t)$ is then used in the computation of σ pre-linkage values $\texttt{plv}_i(t,c) = \mathit{Enc}(\texttt{ls}_i(t), la\_id_i \,\|\, c)$, for $0 \leqslant c < \sigma$. In [CAM16], the encryption is actually done using the Davies-Meyer construction [Pre05], meaning that the cipher's input is XORed with the ciphertext produced as output; however, since this small difference is not relevant for our discussion, we omit the extra XOR in our notation. Finally, every $\texttt{plv}_i(t,c)$ is truncated to a suitable length, individually encrypted and authenticated[1] using a key shared between the PCA and LA_i, , and then sent to the RA. The RA simply includes this encrypted information, together with the corresponding cocoon keys, in the requests sent to the PCA. The latter can subsequently compute the linkage values to be included in the resulting certificates. In the usual case, which consists of two LAs participating in this process, the linkage value for the c-th certificate valid in time period t is computed as $\texttt{lv}(t,c) = \texttt{plv}_1(t,c) \oplus \texttt{plv}_2(t,c)$.

As a result of this process, whenever a device is identified as malicious by a Misbehavior Authority (MA), certificates still valid owned by that device can be revoked not only individually, but also altogether. This is accomplished via the collaboration of the PCA, RA, and LAs. Namely, the PCA can associate the $\texttt{lv}$ informed by the MA to the original pseudonym certificate request received from the RA. The PCA then provides this information, together with the corresponding pre-linkage values $\texttt{plv}_i(t,c)$, to the RA. The RA, in turn, can (1) identify the device behind that certificate request, placing its enrollment certificate in a blacklist for preventing it from obtaining new pseudonym

[1] Even though the authentication of pre-linkage values is not explicitly mentioned in [WWKH13, CAM16], doing so is important to prevent forgery by the RA. Otherwise by delivering fake pre-linkage values to the PCA as if they came from LA_i, a dishonest RA would be able to track devices.

certificates, and (2) ask LA_i to identify the linkage seed $\texttt{ls}_i(0)$ from which $\texttt{plv}_i(t, c)$ was computed. Finally, each LA_i provides the RA with $\texttt{ls}_i(t_s)$, where t_s is the time period from which the revocation starts being valid (usually, the current time period or the one in which the misbehavior was first detected). The set of $\texttt{ls}_i(t_s)$ received from the LAs are sent to the MA, so they can be placed in a CRL to be distributed throughout the system. This allows any entity to compute $\texttt{lv}(t, c)$ for time periods $t \geqslant t_s$, linking the corresponding certificates to a single CRL entry. Consequently, *current* and *future* certificates owned by the misbehaving device are revoked and can be linked to that device; *past* certificates remain protected, though, preserving the device's privacy prior to the detection of the malicious activity.

5 An improved pseudonym certificate provisioning process

Albeit quite efficient, in particular from the vehicles' perspective, SCMS's pseudonym certificate provisioning protocol described in Section 4.1 can be further optimized. Specifically, the butterfly key expansion procedure is executed twice by the RA for each pseudonym certificate: once for the generation of the public signature keys and another for encryption keys. As a result, the device itself needs to send to the RA two caterpillar keys (S and E), as well as the corresponding PRFs (f_s and f_e), for the computation of the corresponding cocoon keys ($\hat{S}_i$ and $\hat{E}_i$, where $0 \leqslant i < \beta$). In addition, since $\hat{S}_i$ and $\hat{E}_i$ are seen as independent keys by the PCA when issuing a certificate, the PCA needs not only to encrypt the certificate but also sign the resulting encrypted package to avoid manipulation by the RA. Even if an efficient signcryption algorithm [Zhe97] is employed for this purpose, the mere existence of this extra signature leads to overheads in multiple places: on the PCA, for the computation and transmission of such signature; on the RA, for its reception and re-transmission; and on the end devices, for its reception and verification, besides the verification of the certificate's signature itself.

It turns out, however, that the generation and usage of encryption and signature keys can be done in a unified manner, leading to better efficiency without loss of security or functionality. The proposed process is described and analyzed in the following subsections. Along the discussion, we assume a security model slightly more powerful than the "honest-but-curious" approach usually adopted in the literature [WWKH13, KP15]. Namely, we consider that the system's entities may be "dishonest-if-allowed": they may engage in active attacks, subverting the protocols if this would bring them some advantage (e.g., the ability to track vehicles); however, we assume such misbehavior occurs only if it can go undetected, so they cannot be held accountable. Therefore, when describing the proposed enhancements, we also discuss how possible deviations from the protocol can be detected.

5.1 Unified butterfly keys

The proposed butterfly key expansion process is depicted in Figure 4, and described as follows.

First, the vehicle generates a single caterpillar private/public key pair $(s, S = s \cdot G)$, and sends S together with a PRF f_s to the RA. The RA then generates β public cocoon signature keys $\hat{S}_i = S + f_s(i) \cdot G$ for several devices, shuffles them, and sends the resulting batch to the PCA.

The subsequent procedure followed by the PCA when processing $\hat{S}_i$ depends on whether explicit or implicit certificates are employed, but it follows the same steps described in Section 4.1. Namely, for implicit certificates the PCA uses $\hat{S}_i$ to generate the device's credential V_i, following the implicit certificates issuing process from [Cer13]. For explicit certificates, the device's public signature key is computed directly from $\hat{S}_i$, by making

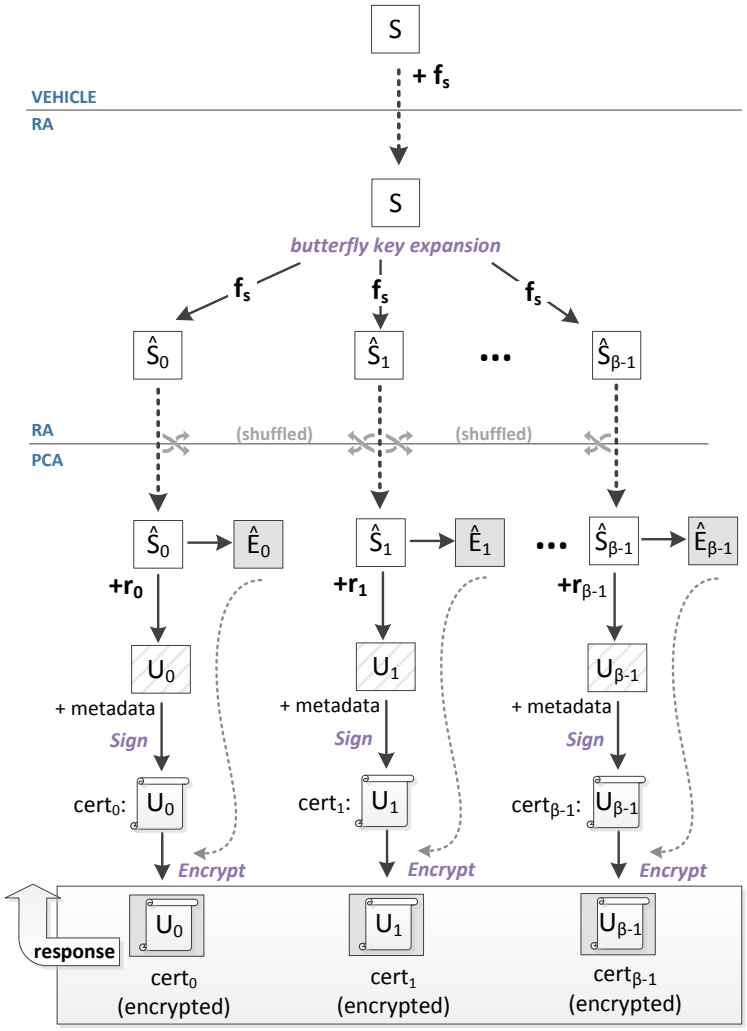

Figure 4: A more efficient, unified butterfly key expansion.

$\mathrm{U}_i = \hat{S}_i + r_i \cdot G$. Whichever the case, the PCA also computes the public cocoon encryption key $\hat{E}_i$ from the received $\hat{S}_i$, by making $\hat{E}_i = Hash(\hat{S}_i) \cdot G + \hat{S}_i$.

After generating the (implicit or explicit) certificate $cert_i$, the PCA uses $\hat{E}_i$ to encrypt its value together with any additional data that needs to be relayed to the vehicle (e.g., r_i for explicit certificates, and sig_i for implicit ones). This encrypted package is then sent to the RA, which in turn delivers it to the vehicle.

The vehicle can then compute the corresponding decryption key $\hat{e}_i = Hash(\hat{S}_i) + s + f_s(i)$ and retrieve $cert_i$. This decryption key works because the encryption is performed using $\hat{E}_i = Hash(\hat{S}_i) \cdot G + \hat{S}_i = Hash(\hat{S}_i) \cdot G + s \cdot G + f_s(i) \cdot G$ as public key. For implicit certificates, U_i is computed and verified as usual, using $cert_i$ and PCA's public signature key. For explicit certificates, the vehicle (1) verifies the PCA's signature on $cert_i$, which encloses U_i, and then (2) computes $\mathrm{u}_i = r_i + s + f_s(i)$ from the value of r_i received in the encrypted package, ascertaining that $\mathrm{u}_i \cdot G = \mathrm{U}_i$.

We note that an analogous process could be adopted for deriving the public cocoon signature key from the public cocoon encryption key if the latter is provided, simply by making $\hat{S}_i = Hash(\hat{E}_i) \cdot G + \hat{E}_i$. For conciseness, however, we omit the details on this alternative approach along the discussion.

Table 2: Issuing pseudonym certificates: comparison between the original SCMS and the proposed solution.

	Vehicle	→	RA	→	PCA	-RA→	Vehicle
SCMS (explicit)	$s, S = s \cdot G$ $e, E = e \cdot G$	$S, E,$ f_s, f_e	$\hat{S}_i = S + f_s(i) \cdot G$ $\hat{E}_i = E + f_e(i) \cdot G$ $(0 \leqslant i < \beta)$	$\hat{S}_i,$ $\hat{E}_i$	$\mathrm{U}_i = \hat{S}_i + r_i \cdot G$ $cert_i = (\mathrm{U}_i, \texttt{meta})$ $sig_i = Sign(u, cert_i)$ $pkg =$ $Sign(u, Enc(\hat{E}_i, \{cert_i, sig_i, r_i\}))$	pkg	$\hat{e}_i = e + f_e(i)$ $\{cert_i, sig_i, r_i\} = Dec(\hat{e}_i, Ver(\mathcal{U}, pkg))$ $Ver(\mathcal{U}, cert_i)$ $\mathrm{u}_i = r_i + s + f_s(i)$
SCMS (implicit)					$V_i = \hat{S}_i + r_i \cdot G$ $cert_i = (V_i, \texttt{meta})$ $sig_i = Hash(cert_i) \cdot r_i + u$ $pkg =$ $Sign(u, Enc(\hat{E}_i, \{cert_i, sig_i\}))$		$\hat{e}_i = e + f_e(i)$ $\{cert_i, sig_i\} = Dec(\hat{e}_i, Ver(\mathcal{U}, pkg))$ $h_i = Hash(cert_i)$ $\mathrm{u}_i = h_i \cdot (s + f_s(i)) + sig_i$ $\mathrm{U}_i = \mathrm{u}_i \cdot G \stackrel{?}{=} h_i \cdot V_i + \mathcal{U}$
ours (explicit)	$s, S = s \cdot G$	S, f_s	$\hat{S}_i = S + f_s(i) \cdot G$ $(0 \leqslant i < \beta)$	$\hat{S}_i$	$\mathrm{U}_i = \hat{S}_i + r_i \cdot G$ $cert_i = (\mathrm{U}_i, \texttt{meta})$ $sig_i = Sign(u, cert_i)$ $\hat{E}_i = Hash(\hat{S}_i) \cdot G + \hat{S}_i$ $pkg = Enc(\hat{E}_i, \{cert_i, sig_i, r_i\})$	pkg	$\hat{e}_i = Hash(\hat{S}_i) + s + f_s(i)$ $\{cert_i, sig_i, r_i\} = Dec(\hat{e}_i, pkg)$ $Ver(\mathcal{U}, cert_i)$ $\mathrm{u}_i = r_i + s + f_s(i)$ $\mathrm{u}_i \cdot G \stackrel{?}{=} \mathrm{U}_i$
ours (implicit)					$V_i = \hat{S}_i + r_i \cdot G$ $cert_i = (V_i, \texttt{meta})$ $sig_i = Hash(cert_i) \cdot r_i + u$ $\hat{E}_i = Hash(\hat{S}_i) \cdot G + \hat{S}_i$ $pkg = Enc(\hat{E}_i, \{cert_i, sig_i\})$		$\hat{e}_i = Hash(\hat{S}_i) + s + f_s(i)$ $\{cert_i, sig_i\} = Dec(\hat{e}_i, pkg)$ $h_i = Hash(cert_i)$ $\mathrm{u}_i = h_i \cdot (s + f_s(i)) + sig_i$ $\mathrm{U}_i = \mathrm{u}_i \cdot G \stackrel{?}{=} h_i \cdot V_i + \mathcal{U}$

Table 2 summarizes and compares the processes adopted in SCMS and in the unified approach when issuing certificates.

5.2 Resulting security against MitM attacks by RA

The overall security of the proposed scheme builds upon the same principles as the original SCMS butterfly key expansion. Namely, a vehicle's caterpillar private key s remains protected by the elliptic curve discrete logarithm problem (ECDLP) during the whole execution of the protocol. Hence, neither RA or PCA is able to recover the signature or decryption private keys derived from it, even if they collude. Unlinkability among certificates is similarly preserved, as long as the RA and PCA do not collude: the shuffling done by the RA still hides from the PCA any relationship between certificate requests intended for a same vehicle; meanwhile, the PCA's encrypted response prevents anyone but the corresponding vehicle from learning $cert_i$.

Hence, the potential security vulnerability introduced by the process hereby proposed refers to the lack of the PCA signature on its encrypted response. Even though this might enable MitM attacks by the RA, the proposed protocol actually prevents such attacks despite the absence of this signature. The reason is that the PCA computes the encryption key $\hat{E}_i$ from $\hat{S}_i$ in such a manner that any manipulation by the RA is detectable by the device when validating the public key U_i, either explicitly or implicitly. Indeed, without loss of generality, suppose that a malicious RA replaces the correct value of $\hat{S}_i$ by $\hat{S}_i^* = z \cdot G$, for an arbitrary value of z. In this case, the PCA ends up encrypting the certificate with $\hat{E}_i = Hash(\hat{S}_i^*) \cdot G + \hat{S}_i^*$. As a result, the RA can decrypt the PCA's response with the decryption key $\hat{e}_i^* = Hash(\hat{S}_i^*) + z$.

For implicit certificates, this attack would allow the RA to learn the vehicle's implicit certificate $cert_i^* = (V_i^*, \texttt{meta})$, where $V_i^* = \hat{S}_i^* + r_i \cdot G$, as well as its corresponding signature $sig_i^* = Hash(cert_i^*) \cdot r_i + u$, where $h_i^* = Hash(cert_i^*)$. However, sig_i^* would not be a valid signature for the actual $\hat{S}_i$ expected by the device. More precisely, after the

vehicle computes $\mathrm{U}_i = \mathrm{u}_i \cdot G$ for $\mathrm{u}_i = h_i^* \cdot (s + f_s(i)) + sig_i^*$, the implicit verification $\mathrm{U}_i \stackrel{?}{=} h_i^* \cdot V_i^* + \mathcal{U}$ fails, unless $z = s + f_s(i)$:

$$\begin{aligned} \mathrm{U}_i &= h_i^* \cdot V_i^* + \mathcal{U} \\ \mathrm{u}_i \cdot G &= h_i^* \cdot (\hat{S}_i^* + r_i \cdot G) + u \cdot G \\ (h_i^* \cdot (s + f_s(i)) + sig_i^*) \cdot G &= (h_i^* \cdot (z + r_i) + u) \cdot G \\ h_i^* \cdot (s + f_s(i)) + h_i^* \cdot r_i + u &= h_i^* \cdot (z + r_i) + u \\ h_i^* \cdot (s + f_s(i)) &= h_i^* \cdot z \\ s + f_s(i) &= z \qquad \triangleright \text{ Assuming } h_i^* \neq 0 \end{aligned}$$

In other words, the implicit signatures generated by the PCA already prevent manipulation of $\hat{S}_i$ by the RA, which indirectly prevents $\hat{E}_i$ itself from being manipulated in MitM attacks.

An analogous argument applies for explicit certificates. In this case, the delivery of $\hat{S}_i^* = z \cdot G$ to the PCA allows the RA to decrypt the PCA's response and learn (1) the device's final public key $\mathrm{U}_i^* = r_i \cdot G + \hat{S}_i^*$ enclosed in the certificate and (2) the value of r_i itself. Before re-encrypting the certificate, however, the RA would have to find a value r_i^* that satisfies $r_i^* \cdot G + \hat{S}_i = \mathrm{U}_i^*$ (Equation 1), so r_i^* can replace the original r_i provided by the PCA in the RA-to-vehicle response. Otherwise, the vehicle would notice that $(r_i^* + s + f_s(i)) \cdot G \neq \mathrm{U}_i^*$. Hence, this final verification would fail and the corresponding key would be identified as invalid by the device, frustrating the attack. Unfortunately for the malicious RA, finding such r_i^* in Equation 1 corresponds to solving the elliptic curve discrete logarithm problem (ECDLP) for the point $(\mathrm{U}_i^* - \hat{S}_i)$ – or, equivalently, it requires computing $r_i^* = (r_i + z) - (s + f_s(i))$ without knowing s. Therefore, the attack itself becomes computationally unfeasible for cryptographically secure elliptic curves.

Notice that the RA might prefer to satisfy Equation 1 by manipulating $\hat{S}_i$ or U_i^* instead of (or in addition to) trying to find a valid r_i^*, but that is not possible in the proposed scheme: U_i^* is signed by the PCA, so any violation of its integrity would be detectable by the end devices upon reception; meanwhile, $\hat{S}_i$ is seen a fixed value on the vehicle's side, so it is not susceptible to modification by the RA.

5.3 Additional security aspects

As an additional remark, the original SCMS design proposes the adoption of two caterpillar keys most likely because it is considered a good practice to avoid using the same (or even correlated) public/private key for encryption and signature. The reasons for this recommendation are basically two [CJNP02]. The first is that eventual vulnerabilities (e.g., implementation errors) found in one process may leak the key for the other. The second is that one process might be used as an oracle for its counterpart: when using algorithms where encryption and signature are correlated procedures (e.g., unpadded RSA [RSA78]), attackers might ask the owner of the key pair to sign some piece of data d with the private key, so the result could serve as the decryption of some ciphertext related to d; analogously, a plaintext obtained via decryption with a private key might be turned into a signature.

At first sight, it may seem that the strategy hereby described violates this general rule by creating a correlation between $\hat{E}_i$ and $\hat{S}_i$. However, in the specific context targeted by the our solution this correlation is not an actual issue, for a few reasons. First, regarding oracles, we note that $\hat{E}_i$ is not employed for encrypting just any known or even predictable piece of data; instead, it is only used once by the PCA, for performing the probabilistic encryption of the vehicle's certificate (or implicitly certified asymmetric key), which is then decrypted, also once, by the vehicle. Hence, it is highly unlikely that this single encryption/decryption process could serve as a useful oracle in practice, especially because (1) vehicles should decrypt only valid certificates and (2) the encryption algorithm recommended for the target scenario [CAM16], ECIES, prevents chosen ciphertext attacks [IEE04]. In addition,

even if a vehicle is somehow tricked into decrypting multiple ciphertext queries, in ECIES the decrypted data obtained has no direct correlation with the private key itself – namely, unlike simple RSA-based schemes, data is encrypted/decrypted and authenticated/verified with a secret symmetric key pair $(\mathcal{K}_E, \mathcal{K}_A)$, not using the private key directly. Therefore, oracle queries are unlikely to leak any information about the private key, nor about values that could be derived from it to be then employed in signature schemes.

Second, and even more importantly, information leaked from the encryption process should not affect the signature process because they actually do not use the same key pair: the encryption is protected by the private key $\hat{e}_i = \mathit{Hash}(\hat{S}_i) + s + f_s(i)$, whereas the device's private signature key is $\mathrm{u}_i = r_i + s + f_s(i)$. In other words, albeit correlated to $\hat{e}_i$, the cocoon signature key $\hat{s}_i = s + f_s(i)$ is not employed for signing any piece of data, at least not before it is further randomized by the PCA using r_i. Therefore, as long as $r_i \neq \mathit{Hash}(\hat{S}_i)$, any predictable correlation between the encryption and the signature processes is eliminated from the perspective of all entities except for the PCA itself. Consequently, even if a flaw in the encryption algorithm allows private keys to be recovered from a limited set of plaintext/ciphertext pairs, that key might only be useful for a dishonest PCA seeking to forge signatures. We can conclude, thus, that the proposed approach copes with the "honest-but-curious" scenario, and should not lead to security issues if a secure encryption algorithm is employed in a "dishonest-if-allowed" scenario.

5.4 Performance analysis

Besides preserving SCMS's security properties, this unified butterfly key expansion leads to a reduced overhead when compared to the original process:

- Vehicle: since the request sent to the RA includes a single cocoon public key and a single PRF rather than two, the processing and bandwidth costs involved in this process drop by half. The batches received are also smaller, because each encrypted package containing a certificate is not signed (only the certificate itself is). Finally, the processing costs for validating the received certificates also decrease, since (1) for implicit certificates, the only difference in computation is that one less signature verification is required, whereas (2) for explicit certificates, vehicles need to perform a single elliptic-curve (EC) multiplication when verifying the received U_i, which is more efficient than a signature verification.

- RA: It only performs the butterfly key expansion for signature keys, leading to half the processing overhead. Ignoring ancillary metadata, bandwidth usage is similarly reduced when forwarding the request to the PCA, which involves a single cocoon key and a single PRF rather than two of each. Finally, the response by the PCA is also smaller due to the absence of a signature on the encrypted package.

- PCA: The extra overhead involved in deriving $\hat{E}_i$ from $\hat{S}_i$ is compensated by the fact that each certificate issuance involves a single signature instead of two. Hence, processing costs when generating implicit or explicit certificates should remain similar. Inbound and outbound bandwidth are saved, though, since the RA's requests are smaller (they do not include $\hat{E}_i$) and so are the PCA's responses (one less signature is sent).

To give some concrete numbers, Table 3 compares the estimated costs of the proposed procedure with the original SCMS as described in [CAM16], assuming the algorithms thereby recommended: ECDSA for signature generation/verification and ECIES for asymmetric encryption/decryption. Both algorithms are configured to provide a 128-bit security level. The performance costs are measured in cycles, using the RELIC cryptography library [AG18] running on a Intel i5 4570 processor. The bandwidth costs are measured

Table 3: Comparison of processing (in cycles, shown in a gray background) and communication (in bytes) costs between the original SCMS and the proposed solution when issuing β certificates, including request and response.

	Vehicle	→	RA	→	PCA	→	RA	→	Vehicle
SCMS (explicit)	514×10^3	96	$\beta\cdot(504\times10^3)$	$\beta\cdot(64)$	$\beta\cdot(3.07\times10^6)$	$\beta\cdot(\lvert cert\rvert+80)$	0	$\beta\cdot(\lvert cert\rvert+80)$	$\beta\cdot(6.67\times10^6)$
ours (explicit)	257×10^3	48	$\beta\cdot(252\times10^3)$	$\beta\cdot(32)$	$\beta\cdot(3.07\times10^6)$	$\beta\cdot(\lvert cert\rvert+48)$	0	$\beta\cdot(\lvert cert\rvert+48)$	$\beta\cdot(4.61\times10^6)$
Ratio (ours/SCMS)	0.5	0.5	0.5	0.5	1.00	[0.75, 1[‡	0	[0.75, 1[‡	0.69
SCMS (implicit)	514×10^3	96	$\beta\cdot(504\times10^3)$	$\beta\cdot(64)$	$\beta\cdot(2.81\times10^6)$	$\beta\cdot(\lvert cert\rvert+48)$	0	$\beta\cdot(\lvert cert\rvert+48)$	$\beta\cdot(6.67\times10^6)$
ours (implicit)	257×10^3	48	$\beta\cdot(252\times10^3)$	$\beta\cdot(32)$	$\beta\cdot(2.81\times10^6)$	$\beta\cdot(\lvert cert\rvert+16)$	0	$\beta\cdot(\lvert cert\rvert+16)$	$\beta\cdot(4.36\times10^6)$
Ratio (ours/SCMS)	0.5	0.5	0.5	0.5	1.00	[0.67, 1[‡	0	[0.67, 1[‡	0.65

‡ The interval is computed ignoring encryption overheads and assuming $\lvert cert\rvert \geqslant 48$, which is expected to be close to the minimum for a certificate (32 bytes for representing U_i or V_i, plus 16 bytes of metadata)

in bytes, ignoring eventual metadata not strictly related to the butterfly key expansion process (e.g., pre-linkage values, time period to which the certificate should be associated, etc.).

6 Conclusions

Data authentication and user privacy are essential for preventing abuse in intelligent transportation systems, either by drivers or by the system itself. This is, however, a challenging task, in particular because any acceptable solution needs to cope with constraints on the vehicle's side such as limited connectivity and processing power. Fortunately, SCMS's pseudonym certificates provisioning and revocation processes are able to address such requirements while also taking into account performance and scalability issues.

Despite those advances, in this article we show that there are still optimization opportunities in the SCMS architecture. Specifically, we describe a novel, unified butterfly key expansion in which two vehicle-provided keys are replaced by a single one. Besides eliminating the need of including such extra key in the vehicle's requests, this approach also removes one signature from each pseudonym certificate generated in response (and, hence, the corresponding costs for their creation, transmission and verification). As a result, when compared to SCMS's pseudonym certificate provisioning protocol, we are able to obtain processing and bandwidth savings (both downstream and upstream) that reach as high as 50%. This is specially relevant when considering that the number of certificates provisioned per vehicle is expected to range from a few thousands [WWKH13] to tens of thousands [KPW17]. In addition, these gains are more noticeable at the vehicles' side, which are exactly the most resource-constrained entities in the system. Finally, the proposed schemes works for either implicit or explicit certificates, while still preserving the system's security, flexibility and scalability in both cases.

As future works, we plan to investigate further optimizations in SCMS's design and related solutions. In special, we are interested in searching for potential synergies between the unified butterfly key expansion process hereby proposed and BCAM's strategy for the delayed activation of pseudonym certificates. Another open issue worth investigating is the possibility of raising the collusion resistance of the system, so three or more entities would have to collude before being able to track devices. This seems to be a particularly challenging issue, however, because RA and PCA together concentrate a great deal of information about vehicles, and this information can be considered essential for their operation in the SCMS architecture.

Thanks

This work was supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under grant 305350/2013-7, and by LG Electronics via the Foundation for the Technological Development of the Engineering Sciences (FDTE).

References

[AG18] D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`, 2018.

[AGMM15] K. Alheeti, A. Gruebler, and K. McDonald-Maier. An intrusion detection system against malicious attacks on the communication network of driverless cars. In *12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 916–921, Jan 2015.

[BDL+12] D. Bernstein, N. Duif, T. Lange, P. Schwabe, and B-Y. Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, Sep 2012. See also `http://ed25519.cr.yp.to/eddsa-20150704.pdf`.

[CAM16] CAMP. Security credential management system proof–of–concept implementation – EE requirements and specifications supporting SCMS software release 1.1. Technical report, Vehicle Safety Communications Consortium, may 2016.

[Cer13] Certicom. Sec 4 v1.0: Elliptic curve Qu-Vanstone implicit certificate scheme (ECQV). Technical report, Certicom Research, 2013. Available: `http://www.secg.org/sec4-1.0.pdf`.

[CJNP02] Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier. Universal padding schemes for RSA. In *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'02)*, pages 226–241, London, UK, UK, 2002. Springer-Verlag.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In *Advances in Cryptology — EUROCRYPT '91: Workshop on the Theory and Application of Cryptographic Techniques Brighton, UK, April 8–11, 1991 Proceedings*, pages 257–265, Berlin, Heidelberg, 1991. Springer.

[Dou02] J. Douceur. The Sybil attack. In *Proc. of 1st International Workshop on Peer-to-Peer Systems (IPTPS)*. Springer, January 2002.

[FKL14] D. Förster, F. Kargl, and H. Löhr. PUCA: A pseudonym scheme with user-controlled anonymity for vehicular ad-hoc networks (VANET). In *IEEE Vehicular Networking Conference (VNC)*, pages 25–32, Dec 2014.

[HPY+14] J. Harding, G. Powell, R. Yoon, J. Fikentscher, C. Doyle, D. Sade, M. Lukuc, J. Simons, and J. Wang. Vehicle-to-vehicle communications: Readiness of V2V technology for application. Technical Report DOT HS 812 014, National Highway Traffic Safety Administration, Washington, DC, USA, 2014.

[IEE04] IEEE. *IEEE Standard Specifications for Public-Key Cryptography – Amendment 1: Additional Techniques*. IEEE Computer Society, 2004.

[IKRK08] A. Iyer, A. Kherani, A. Rao, and A. Karnik. Secure V2V communications: Performance impact of computational overheads. In *IEEE INFOCOM Workshops*, pages 1–6, April 2008.

[JD08] D. Jiang and L. Delgrossi. IEEE 802.11p: Towards an international standard for wireless access in vehicular environments. In *IEEE Vehicular Technology Conference (VTC Spring)*, pages 2036–2040, 2008.

[JL17] S. Josefsson and I. Liusvaara. RFC 8032 – edwards-curve digital signature algorithm (EdDSA). `https://tools.ietf.org/html/rfc8032`, January 2017.

[KP15] M. Khodaei and P. Papadimitratos. The key to intelligent transportation: Identity and credential management in vehicular communication systems. *IEEE Vehicular Technology Magazine*, 10(4):63–69, Dec 2015.

[KPW17] Virendra Kumar, Jonathan Petit, and William Whyte. Binary hash tree based certificate access management for connected vehicles. In *Proc. of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec'17, pages 145–155, New York, NY, USA, 2017. ACM.

[Lam81] L. Lamport. Password authentication with insecure communication. *Commun. ACM*, 24(11):770–772, 1981.

[MLHS12] R. Moalla, B. Lonc, H.Labiod, and N. Simoni. Risk analysis study of ITS communication architecture. In *3rd International Conference on The Network of the Future*, pages 2036–2040, 2012.

[NIS01] NIST. *Federal Information Processing Standard (FIPS 197) – Advanced Encryption Standard (AES)*. National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD, USA, November 2001.

[NIS13] NIST. *Federal Information Processing Standard (FIPS 186-4) – Digital Signature Standard (DSS)*. National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD, USA, July 2013.

[NIS15a] NIST. *Federal Information Processing Standard (FIPS 180-4) – Secure Hash Standard (SHS)*. National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD, USA, August 2015. DOI:10.6028/NIST.FIPS.180-4.

[NIS15b] NIST. *Federal Information Processing Standard (FIPS 202) – SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD, USA, August 2015. DOI:10.6028/NIST.FIPS.202.

[PFE+09] P. Papadimitratos, A. La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza. Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation. *IEEE Communications Magazine*, 47(11):84–95, November 2009.

[Pre05] Bart Preneel. *Davies–Meyer Hash Function*, pages 136–136. Springer US, Boston, MA, 2005.

[PSFK15] J. Petit, F. Schaub, M. Feiri, and F. Kargl. Pseudonym schemes in vehicular networks: A survey. *IEEE Communications Surveys Tutorials*, 17(1):228–255, 2015.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology: Proceedings of CRYPTO'84*, pages 47–53, Berlin, Heidelberg, Aug 1985. Springer.

[SMK09] F. Schaub, Z. Ma, and F. Kargl. Privacy requirements in vehicular communication systems. In *Proc. of the International Conference on Computational Science and Engineering*, volume 3, pages 139–145. IEEE, 2009.

[Ver16] Eric Verheul. Activate later certificates for V2X – combining ITS efficiency with privacy. Cryptology ePrint Archive, Report 2016/1158, 2016.

[WWKH13] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn. A security credential management system for V2V communications. In *IEEE Vehicular Networking Conference*, pages 1–8, 2013.

[ZAFB12] S. Zhao, A. Aggarwal, R. Frost, and X. Bai. A survey of applications of identity-based cryptography in mobile ad-hoc networks. *IEEE Communications Surveys Tutorials*, 14(2):380–400, Feb 2012.

[Zhe97] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) « cost(signature) + cost(encryption). In *Advances in Cryptology — CRYPTO '97: 17th Annual International Cryptology Conference*, pages 165–179. Springer Berlin Heidelberg, 1997.