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 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 broadcast. Such concerns create the need for privacy-preserving mechanisms to protect the drivers' privacy: even though this is unlikely to prevent vehicles from being tracked by traditional means (e.g., via video surveillance), at least the underlying V2X architecture itself should not be prone to abuse.

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 sent back to the RA, from which they are 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 8 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
U,$\mathcal{U}$	Public signature keys (stylized $\mathcal{U}$: reserved for PCA)
u,u	Private keys corresponding to U and $\mathcal{U}$, respectively
S, s	Public and private caterpillar signature keys, respectively
E, e	Public and private caterpillar encryption keys, respectively
$\hat{S}, \hat{s}$	Public and private cocoon signature keys, respectively
$\hat{E}, \hat{e}$	Public and private cocoon encryption keys, respectively
X, x	Public and private unified caterpillar keys, respectively
$\widehat{X}, \hat{x}$	Public and private unified cocoon keys, respectively
β	Number of cocoon keys in pseudonym certificates batch
la_id	ID of a Linkage Authority (LA)
$\texttt{ls}_i$	Linkage seed
$\texttt{plv}_i$	Pre-linkage value
`lv`	Linkage value
τ	Number of time periods covered by pseudonym certificates 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 the 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 the 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.

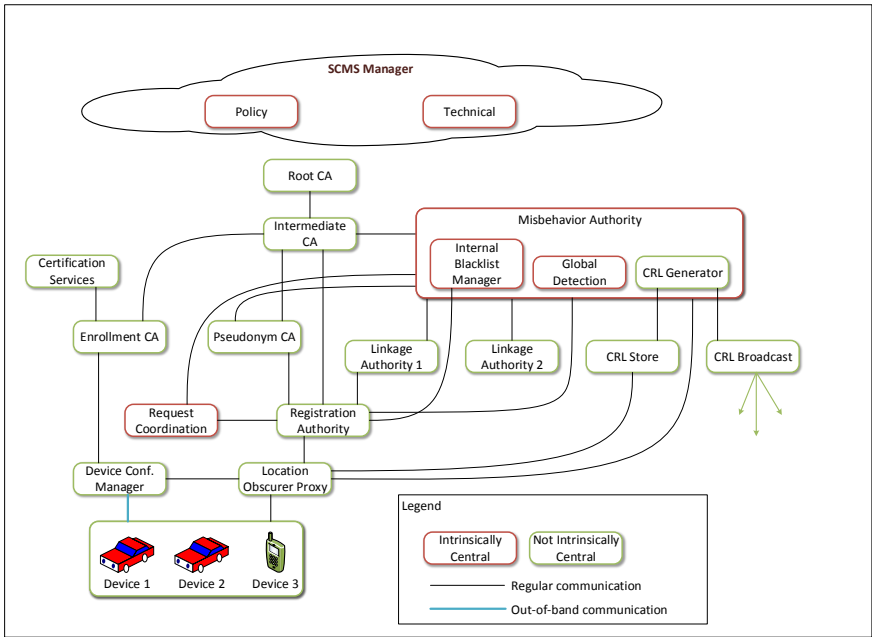

Figure 1: SCMS overview. Source:[WWKH13].

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.

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_1 and f_2 (or, more precisely, to their corresponding seeds). The key S is employed by the RA in the generation of β public *cocoon* signature keys $\hat{S}_i = S + f_1(i) \cdot G$, where $0 \leqslant i < \beta$ for an arbitrary value of β; similarly, the RA uses E for generating β public cocoon encryption keys $\hat{E}_i = E + f_2(i) \cdot G$. The exact manner by which f_1 and f_2 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 an 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. The PCA's response is then sent to the RA, which relays it (in batch) to the requesting vehicle. As are result of this process, only the corresponding vehicle can decrypt the PCA's responses to learn U_i and compute the

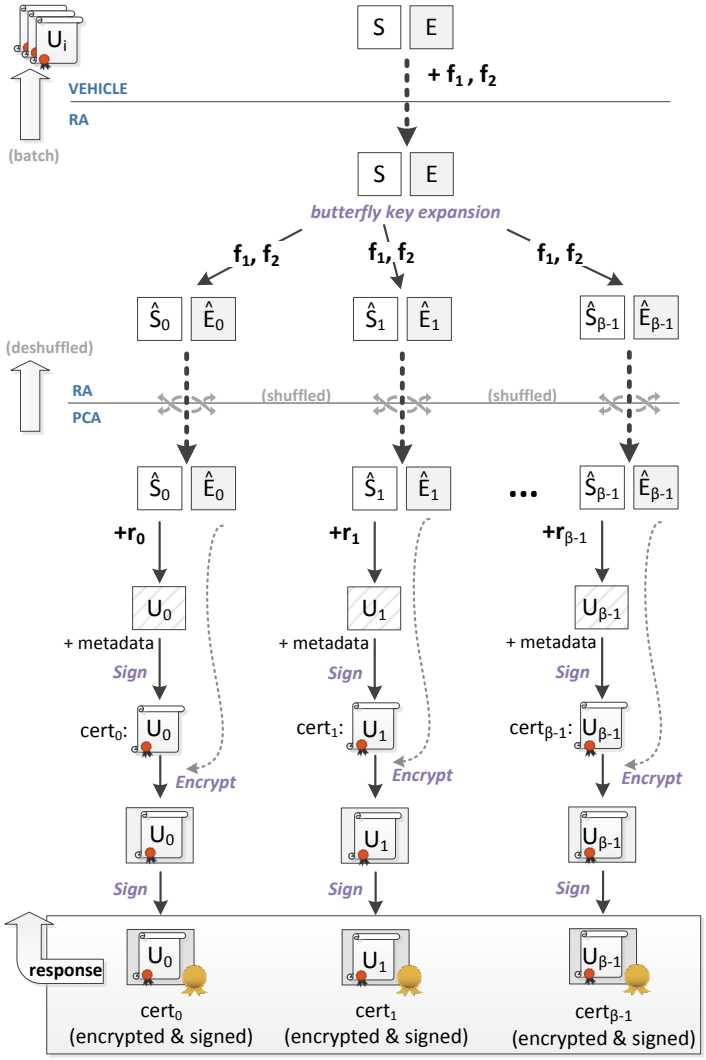

Figure 2: SCMS's butterfly key expansion and (explicit) certificate generation.

corresponding private signature key $u_i = s + r_i + f_1(i)$. To ensure this is the correct key, the vehicle should also perform a final verification $u_i \cdot G \stackrel{?}{=} U_i$.

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 (via the RA) the pair $(cert_i, sig_i)$ encrypted with $\hat{E}_i$. The vehicle, after decrypting the PCA's response, computes $h_i = Hash(cert_i)$ and sets its own private signature key to $u_i = h_i \cdot (s + f_1(i)) + sig_i$, 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

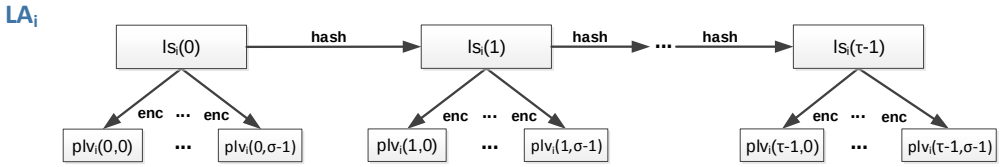

Figure 3: SCMS's key linkage process: generation, by LA_i, of pre-linkage values employed for certificate revocation.

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 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 the 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 the 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 $\mathtt{lv}$, computed by XORing ℓ pre-linkage values $\mathtt{plv}_i$ (where $1 \leqslant i \leqslant \ell$) provided by $\ell \geqslant 2$ different Linkage Authorities (LA). The generation of $\mathtt{plv}_i$ 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 $\mathtt{ls}_i(0)$. Then, if the RA's request covers τ certificate time periods, LA_i iteratively computes a τ-long hash chain [Lam81] $\mathtt{ls}_i(t) = \mathit{Hash}(\mathit{la\_id}_i \,\|\, \mathtt{ls}_i(t-1))$, where $\mathit{la\_id}_i$ is LA_i's identity string and $1 \leqslant t < \tau$. Each $\mathtt{ls}_i(t)$ is then used in the computation of σ pre-linkage values $\mathtt{plv}_i(t,c) = \mathit{Enc}(\mathtt{ls}_i(t), \mathit{la\_id}_i \,\|\, c)$, for $0 \leqslant c < \sigma$. Finally, every $\mathtt{plv}_i(t,c)$ 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 $\mathtt{lv}(t,c) = \mathtt{plv}_1(t,c) \oplus \mathtt{plv}_2(t,c)$.

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 $\mathtt{lv}$ informed by the MA to the original pseudonym certificate request received from the RA. The PCA then provides

[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 malicious RA would be able to track devices.

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 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 Improving the key expansion process: Unified butterfly keys

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_1 and f_2), 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 unified butterfly key expansion process is depicted in Figure 4, and described as follows.

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

The subsequent procedure followed by the PCA when processing $\widehat{X}_i$ 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 $\widehat{X}_i$ 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 $\widehat{X}_i$, by making $\text{U}_i = \widehat{X}_i + r_i \cdot G$. Whichever the case, the PCA also sets the package's public encryption key as $\widehat{X}_i$ itself.

After generating the (implicit or explicit) certificate $cert_i$, the PCA uses $\widehat{X}_i$ 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 $\hat{x}_i = x + f(i)$ and

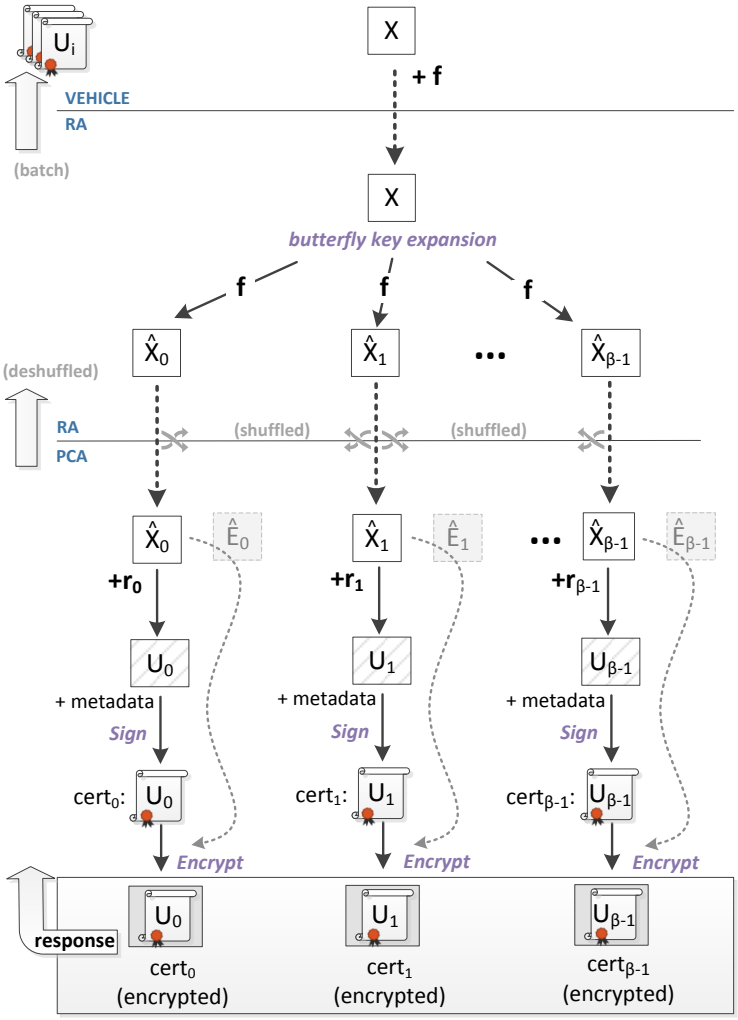

Figure 4: A more efficient, unified butterfly key expansion.

retrieve $cert_i$. This decryption key works because the encryption is performed using $\widehat{X}_i = x \cdot G + f(i) \cdot G$ 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 $u_i = r_i + x + f(i)$ from the value of r_i received in the encrypted package, ascertaining that $u_i \cdot G = U_i$. Whereas this final verification on the received U_i was only advisable in SCMS, for verifying the correctness of the PCA's response, it is mandatory in the proposed approach for avoiding MitM attacks by the RA (as further discussed in Section 6.3).

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

6 Security analysis

The overall security of the proposed scheme builds upon the same principles as the original SCMS butterfly key expansion. Namely, there is no modification on how the process that defines how the caterpillar and cocoon signature keys are handled by PCA and RA. Therefore, the security arguments of SCMS, which rely basically on the fact that x remains

Table 2: Issuing pseudonym certificates: comparison between the original SCMS and the proposed solution. Operations in gray background correspond to the main bandwidth and processing savings resulting from the unified butterfly key approach.

	Vehicle	→	RA	→	PCA	-RA→	Vehicle
SCMS (explicit)	s, $S = s \cdot G$	S, f_1	$\hat{S}_i = S + f_1(i) \cdot G$	$\hat{S}_i$,	$\mathrm{U}_i = \hat{S}_i + r_i \cdot G$ $sig_i = Sign(u, \{\mathrm{U}_i, \texttt{meta}\})$ $cert_i = \{\mathrm{U}_i, \texttt{meta}, sig_i\}$ $pkg = Enc(\hat{E}_i, \{cert_i, r_i\})$ $res = \{pkg, Sign(u, pkg)\}$	res	$\hat{e}_i = e + f_2(i)$ $Ver(\mathcal{U}, res)$ $\{cert_i, r_i\} = Dec(\hat{e}_i, pkg)$ $Ver(\mathcal{U}, cert_i)$ $\mathrm{u}_i = s + f_1(i) + r_i$ $\mathrm{u}_i \cdot G \stackrel{?}{=} \mathrm{U}_i$
SCMS (implicit)	e, $E = e \cdot G$	E, f_2	$\hat{E}_i = E + f_2(i) \cdot G$ $(0 \leqslant i < \beta)$	$\hat{E}_i$	$V_i = \hat{S}_i + r_i \cdot G$ $cert_i = \{V_i, \texttt{meta}\}$ $sig_i = Hash(cert_i) \cdot r_i + u$ $pkg = Enc(\hat{E}_i, \{cert_i, sig_i\})$ $res = \{pkg, Sign(u, pkg)\}$		$\hat{e}_i = e + f_2(i)$ $Ver(\mathcal{U}, res)$ $\{cert_i, sig_i\} = Dec(\hat{e}_i, pkg)$ $h_i = Hash(cert_i)$ $\mathrm{u}_i = h_i \cdot (s + f_1(i)) + sig_i$ $\mathrm{U}_i = \mathrm{u}_i \cdot G \stackrel{?}{=} h_i \cdot V_i + \mathcal{U}$
ours (explicit)	x, $X = x \cdot G$	X, f	$\widehat{X}_i = X + f(i) \cdot G$ $(0 \leqslant i < \beta)$	$\widehat{X}_i$	$\mathrm{U}_i = \widehat{X}_i + r_i \cdot G$ $sig_i = Sign(u, \{\mathrm{U}_i, \texttt{meta}\})$ $cert_i = \{\mathrm{U}_i, \texttt{meta}, sig_i\}$ $pkg = Enc(\widehat{X}_i, \{cert_i, r_i\})$	pkg	$\hat{x}_i = x + f(i)$ $\{cert_i, r_i\} = Dec(\hat{x}_i, pkg)$ $Ver(\mathcal{U}, cert_i)$ $\mathrm{u}_i = \hat{x}_i + r_i$ $\mathrm{u}_i \cdot G \stackrel{?}{=} \mathrm{U}_i$
ours (implicit)					$V_i = \widehat{X}_i + r_i \cdot G$ $cert_i = \{V_i, \texttt{meta}\}$ $sig_i = Hash(cert_i) \cdot r_i + u$ $pkg = Enc(\widehat{X}_i, \{cert_i, sig_i\})$		$\hat{x}_i = x + f(i)$ $\{cert_i, sig_i\} = Dec(\hat{x}_i, pkg)$ $h_i = Hash(cert_i)$ $\mathrm{u}_i = h_i \cdot (x + f(i)) + sig_i$ $\mathrm{U}_i = \mathrm{u}_i \cdot G \stackrel{?}{=} h_i \cdot V_i + \mathcal{U}$

protected by the elliptic curve discrete logarithm problem (ECDLP, given in Definition 1) during the whole execution of the protocol, remain valid. 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 the same vehicle; meanwhile, the PCA's encrypted response prevents anyone but the owner of the decryption key from learning $cert_i$.

Definition 1. Elliptic Curve Discrete Logarithm Problem (ECDLP) [LS08]. Let E be an elliptic curve over a finite field K. Suppose there are points $P, Q \in E(K)$ given such that $Q \in \langle P \rangle$. Determine k such that $Q = k \cdot P$.

On the other hand, the unified key approach introduces two changes to SCMS: (1) it modifies the manner by which the encryption key is computed, and (2) it eliminates the PCA's signature on the encrypted package. The first modification could affect the confidentiality of the communication, thus allowing the RA to learn $cert_i$. Meanwhile, since the final signature made by the PCA on its response is aimed at ensuring the system's security against MitM attacks by the RA, the second modification could result in vulnerabilities on that aspect. However, in what follows we show that the unified key approach still protects the pseudonym certificates' contents and prevents MitM attacks, assuming the hardness of the ECDLP. More precisely, we show that the problem of decrypting the PCA's response encrypted with X can be reduced to an instance of ECDLP.

The same computational hardness applies to MitM attacks, for which we show that the PCA's response is handled in such as manner that any manipulation by the RA is detectable by the device when validating the public key U_i, either explicitly or implicitly.

6.1 Confidentiality of pseudonym certificates

In SCMS, the goal of encrypting the response package with a public encryption key $\hat{E}$ is to prevent the RA from learning its contents. This is accomplished simply by using $\hat{e}$ for which the corresponding private key $\hat{e}$ remains unknown by the RA. What the unified approach hereby proposed does is to build upon the observation that both the encryption $\hat{e}$ and signature $\hat{s}$ private keys need to remain protected in SCMS, which can still be done if they are combined into a single piece of information. Indeed, the security of using $\widehat{X}_i$ directly as encryption key can be verified in the following Theorem 1.

Theorem 1. Security of the unified butterfly key expansion against eavesdropping by RA*: Suppose that the RA follows a honest-but-curious security model, sending the correct $\widehat{X} = X + f(i) \cdot G$ to the PCA. In that case, the RA is unable to recover the contents of the PCA's encrypted response pkg in polynomial time unless it is able to solve an instance of the elliptic curve discrete logarithm problem (ECDLP) in polynomial time.*

Proof. The proof in this case is actually quite straightforward: if the encryption is performed with a secure algorithm, there should be no polynomial-time algorithm that allows decryption without knowledge of $\hat{x}$, nor a polynomial-time algorithm that allows the recovery of this key from *pkg*. Hence, violating the confidentiality of the scheme requires the recovery of $\hat{x}$ from either X or $\widehat{X}$. After all, besides *pkg* itself, these are the only pieces of information possessed by the RA that carry some relationship with the decryption key $\hat{x}$. However, since $\widehat{X} = X + f(i) \cdot G$, where $f(i)$ is known by the RA, this task is equivalent to finding x from X, i.e., to solving the ECDLP for X. □

6.2 Security against MitM attacks by RAs in the implicit model

The security result obtained in Section 6.1 assumes that the RA follows the unified key expansion protocol, providing the correct $\widehat{X}$ to the PCA. However, the RA might prefer to replace this key with $\widehat{X}_i^* = z \cdot G$, for an arbitrary value of z. In this case, the confidentiality of the process would be lost, because the PCA would end up encrypting *pkg* with $\widehat{X}_i^*$ and the result would be trivially decrypted by the RA using the corresponding private key z. Therefore, we need to also consider the security of this Man-in-the-Middle scenario, which is complementary to the "honest-but-curious" scenario previously assumed. We impose no constraint on the (mis)behavior of the RA, letting it freely choose $\widehat{X}_i^*$ as long as the choice: (1) leads to some advantage to the RA, in particular the ability to violate the confidentiality or integrity of *pkg*; and (2) the misbehavior is not detected by vehicles, so they believe it is safe to use the corresponding certificates. With this scenario in mind, we can formulate Theorem 2.

Theorem 2. Security of the unified butterfly key expansion against MitM attacks in the implicit model*: Suppose that the RA replaces $\widehat{X}_i$ by an arbitrary $\widehat{X}_i^*$ in the request for implicit certificates sent to the PCA. Assuming the hardness of the ECDLP and the random oracle model, the RA cannot violate the integrity or confidentiality of the PCA's response pkg without the requesting vehicle's knowledge.*

Proof. We start by noticing that the integrity of *pkg*'s contents is protected despite the lack of the PCA signature over it. Indeed, even if the RA is somehow able to violate the confidentiality of *pkg*, it would only be able to obtain the (signed) implicit certificate $cert_i$. However, $cert_i$ is not treated as confidential in the implicit certification model

[Cer13, Section 3.4], and yet such model ensures the integrity of $cert_i$ in the random oracle model assuming the hardness of the ECDLP [**?**]. Therefore, the implicit certification itself already ensures that any modification of $cert_i$, either directly (i.e., after decrypting pkg) or indirectly (i.e., by modifying only the ciphertext), would be detectable by vehicles.

Proving the confidentiality of the unified key expansion, however, requires some more effort because we cannot rely so directly on the security properties of implicit certificates. Once again, we follow the reductionist approach, showing that violating the confidentiality of pkg requires the resolution of an instance of the ECDLP.

Suppose that the malicious RA replaces the correct value of $\widehat{X}_i$ by $\widehat{X}_i^* = z \cdot G$, for an arbitrary value of z . This assumption comes without loss of generality, since in principle we do not impose any restriction on the actual value of z chosen by the RA. Upon reception of the RA's request, the PCA ends up encrypting the implicit certificate $cert_i$ with $\widehat{X}_i^*$, since it is unable to detect such misbehavior. As a result, the RA can decrypt the PCA's response using z as the decryption key, thus violating the confidentiality of the system. This attack would allow the RA to learn the vehicle's implicit certificate $cert_i^* = (V_i^*, \texttt{meta})$, where $V_i^* = \widehat{X}_i^* + r_i \cdot G$, as well as its corresponding signature $sig_i^* = Hash(cert_i^*) \cdot r_i + u$, where $h_i^* = Hash(cert_i^*)$.

However, this misbehavior by the RA can be detected by the vehicle because, for any $z \neq x + f(i)$, the resulting sig_i^* would not be a valid signature for the actual $\widehat{X}_i$ expected by the vehicle. More precisely, after the vehicle computes $\mathrm{U}_i = \mathrm{u}_i \cdot G$ for $\mathrm{u}_i = h_i^* \cdot (x + f(i)) + sig_i^*$, the implicit verification $\mathrm{U}_i \stackrel{?}{=} h_i^* \cdot V_i^* + \mathcal{U}$ fails, unless $z = x + f(i)$:

$$\begin{aligned}
\mathrm{U}_i &\stackrel{?}{=} h_i^* \cdot V_i^* + \mathcal{U} \\
\mathrm{u}_i \cdot G &\stackrel{?}{=} h_i^* \cdot (\widehat{X}_i^* + r_i \cdot G) + u \cdot G \\
(h_i^* \cdot (x + f(i)) + sig_i^*) \cdot G &\stackrel{?}{=} (h_i^* \cdot (z + r_i) + u) \cdot G \\
h_i^* \cdot (x + f(i)) + h_i^* \cdot r_i + u &\stackrel{?}{=} h_i^* \cdot (z + r_i) + u \\
h_i^* \cdot (x + f(i)) &\stackrel{?}{=} h_i^* \cdot z \\
x + f(i) &\stackrel{?}{=} z \qquad \triangleright \text{ Assuming } h_i^* \neq 0
\end{aligned}$$

Therefore, to be able to bypass the vehicle's verification, the RA cannot just choose any z. Instead, it is obliged to make $z = x + f(i)$. Even though $f(i)$ is known by the RA, finding the value of x that allows the computation of z with this characteristic is equivalent to solving the ECDLP for X.

□

6.3 Security against MitM attacks by RAs in the explicit model

The security arguments for explicit certificates are similar to those presented in Section 6.2 for the implicit model, as summarized in Theorem 3.

Theorem 3. Security of the unified butterfly key expansion against MitM attacks in the implicit model*: Suppose that the RA replaces $\widehat{X}_i$ by an arbitrary $\widehat{X}_i^*$ in the request for explicit certificates sent to the PCA. Assuming the hardness of the ECDLP, the RA cannot violate the integrity or confidentiality of the PCA's response pkg without the requesting vehicle's knowledge.*

Proof. Once again, it is easy to show that the explicit certificate $cert_i$ enclosed in the PCA's encrypted response, pkg, cannot be modified while avoiding detection by vehicles. After all, the $cert_i$ is itself digitally signed by the PCA, so any modification would invalidate the signature assuming that a secure algorithm was employed for its computation. Therefore, even if the confidentiality of pkg is somehow violated by the RA, that might allow the (unsigned) value of r_i to be modified, but not the modification of the (signed) $cert_i$. Indirectly, however, the non malleability of $cert_i$ also ensures that a possible modification

of r_i would be detectable by the vehicle. The reason is that the value of U_i obtained from $cert_i$ is verified by the vehicle when it computes $\mathrm{u}_i = r_i + x + f(i)$ and then checks if $\mathrm{u}_i \cdot G \stackrel{?}{=} \mathrm{U}_i$. Since x and $f(i)$ are known by the vehicle (i.e., cannot be manipulated by the RA), and U_i is fixed in the certificate, turning r_i into $r_i^* \neq r_i$ would lead to $\mathrm{u}_i^* = r_i^* + x + f(i) \neq \mathrm{u}_i$ and hence to $\mathrm{u}_i^* \cdot G \neq \mathrm{U}_i$. Therefore, none of the *pkg*'s contents can be modified without detection by the vehicle.

The final verification performed by the vehicle also ensures the confidentiality of the unified key expansion, assuming the hardness of the ECDLP to which this problem can be reduced. To prove this, we once again suppose without loss of generality that the malicious RA replaces $\widehat{X}_i$ by $\widehat{X}_i^* = z \cdot G$, for an arbitrarily chosen value of z. In this case, the RA uses z to decrypt the PCA's response and then learns: (1) the device's final public key $\mathrm{U}_i^* = r_i \cdot G + \widehat{X}_i^*$ enclosed in the certificate; and (2) the value of r_i itself.

To avoid detection, the RA would then have to re-encrypt the PCA's response in such a manner that the vehicle does not notice that $\widehat{X}_i$ was not used in the computation of the received U_i^*. Accomplishing this requires replacing the original r_i by some r_i^* that passes the verification process performed at the vehicle, i.e., that satisfies $(r_i^* + x + f(i)) \cdot G \stackrel{?}{=} \mathrm{U}_i^*$. Otherwise, the vehicle that performs this final verification would identify the received U_i^* as invalid, frustrating the attack. Unfortunately for the RA, however, this means that r_i^* must be set to $(r_i + z) - (x + f(i))$, meaning that finding such r_i^* is equivalent to solving the ECDLP for the point $(\mathrm{U}_i^* - \widehat{X}_i)$. Equivalently, since $f_1(i)$ is known by the RA, z can be freely chosen by it, and r_i is learned due to the attack, this problem can be reduced to finding x given the value of X provided by the vehicle. Nevertheless, this is still an ECDLP instance, which concludes the proof.

□

6.4 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 pair for encryption and signature. The main reason for this recommendation is that possible vulnerabilities (e.g., implementation errors) found in one process may leak the key for the other [CJNP02]. Hence, if an attacker can somehow interact with a vehicle in such a manner that the latter functions as a decryption (resp. signature) oracle, and then recover the private key employed in this process, that would also give away the private key for the other process.

At first sight, it may seem that the strategy hereby described violates this general rule by creating a key $\widehat{X}_i$ that is used both for encryption (by the PCA) and for generating digital signature (by the vehicles). However, this is not the case in the proposed scheme, because the private key $\hat{x}_i$ corresponding to $\widehat{X}_i$ is actually never used for signing any piece of data. Instead, vehicles use $\mathrm{u}_i = \hat{x}_i + r_i$ as signature keys in the explicit model, and $h_i \cdot \hat{x}_i + sig_i$ in the implicit model, where r_i and sig_i are secret values known only by the vehicle and the PCA. As long as $r_i \neq 0$ (for explicit certificates) and $sig_i \neq 0$ (for implicit ones), any predictable correlation between the encryption and the signature processes is eliminated from the perspective of all entities (as expected from randomly-generated keys), except for the PCA itself. Therefore, recovering $\hat{x}_i$ from a compromised u_i should only be feasible by the PCA, but the latter would gain nothing by doing so because it already knows the plaintext protected with $\hat{x}_i$: after all, the PCA is the one that encrypted that plaintext in the first place. Actually, even if the PCA decides to extend the attack by collaborating with the RA, using the latter's knowledge of $f(i)$ to recover x_i from $\hat{x}_i = x_i + f(i)$, that would only give the colluding parties the ability to track vehicles. Nonetheless, like in the original SCMS, the unified butterfly key expansion does not prevent colluding PCA and RA from tracking vehicles even without any compromise of u_i.

The only potentially useful attack against the unified key expansion process is, thus, to recover u_i from a compromised $\hat{x}_i$, which allows the forgery of messages as if they were signed by the owner of $cert_i$. More precisely, since an attacker that compromises $\hat{x}_i$ also gains access to r_i and sig_i from the PCA's response, the subsequent recovery of u_i is not exclusively feasible by the PCA, at least in principle. We argue, however, that the recovery of $\hat{x}_i$ itself is unlikely to succeed in the scenario targeted by the unified keys, for a few reasons:

1. By design, vehicles are not very useful as decryption oracles in the target scenario. This is due to the fact that, unlike signature keys, each decryption key $\hat{x}_i$ is expected to be employed only *once* by vehicles, for the decryption of a single certificate. The decryption of a second certificate with the same key might occur if the first one was considered invalid, but that would naturally be perceived as indication of a defective or malicious sender. Hence, mounting successful attacks against the encryption procedure performed during certificate issuance is actually much harder than directly attacking the signature process afterward.

2. Even if the attacker is able to bypass the previous restriction, convincing vehicles to reuse a decryption key, they would detect malformed queries typical of (adaptive) chosen-ciphertext attacks, since the plaintexts obtained after decryption must contain valid certificates. In other words, if the device finds out that the decryption result is not a PCA-signed certificate, this would once again indicate a defective or malicious sender. Therefore, decryption queries not sent by the PCA will always lead to a rejection message, unless the attacker is able to forge the PCA's signature Assuming a secure signature algorithm, attackers cannot obtain any useful information from such queries, whereas an insecure signature scheme could be attacked directly instead of via the encryption process. The PCA, in turn, would be restricted to performing chosen-plaintext attacks, i.e., it would have to provide actual certificates in its queries to obtain something other than "invalid" as response (and, also, to avoid being reported as defective/malicious). By doing so, however, the PCA goes back to restriction **??**.

In summary, this means that the proposed approach remains secure except in a scenario in which (1) there is a catastrophic flaw in the encryption scheme that allows chosen plaintext attacks involving a single query and (2) a dishonest PCA engages in such attacks. The resulting attack surface is, thus, very small, and negligible if we assume that the PCA follows the honest-but-curious security model.

7 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 is smaller than in SCMS, since the verification of the PCA's signature on the encrypted package is eliminated. This is particularly interesting for digital signature schemes such as ECDSA, for which verification procedure is usually more expensive than signing messages [BDL+11].

- RA: It only performs the butterfly key expansion for signature keys, leading to half the processing overhead. Ignoring ancillary metadata, bandwidth usage is similarly

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 (RP)*	Vehicle (FP)*
SCMS (explicit)	508×10^3	96	$\beta\cdot(499\times10^3)$	$\beta\cdot(64)$	$\beta\cdot(3.27\times10^6)$	$\beta\cdot(\lvert cert\rvert+80)$	0	$\beta\cdot(\lvert cert\rvert+80)$	$\beta\cdot(5.30\times10^6)$	$\beta\cdot(3.23\times10^6)$
ours (explicit)	254×10^3	48	$\beta\cdot(250\times10^3)$	$\beta\cdot(32)$	$\beta\cdot(2.86\times10^6)$	$\beta\cdot(\lvert cert\rvert+48)$	0	$\beta\cdot(\lvert cert\rvert+48)$	$\beta\cdot(3.75\times10^6)$	$\beta\cdot(2.73\times10^6)$
Ratio: ours/SCMS	0.5	0.5	0.5	0.5	0.88	[0.75, 1[‡	0	[0.75, 1[‡	0.71	0.85
SCMS (implicit)	508×10^3	96	$\beta\cdot(499\times10^3)$	$\beta\cdot(64)$	$\beta\cdot(2.86\times10^6)$	$\beta\cdot(\lvert cert\rvert+48)$	0	$\beta\cdot(\lvert cert\rvert+48)$	$\beta\cdot(5.74\times10^6)$	$\beta\cdot(4.72\times10^6)$
ours (implicit)	254×10^3	48	$\beta\cdot(250\times10^3)$	$\beta\cdot(32)$	$\beta\cdot(2.46\times10^6)$	$\beta\cdot(\lvert cert\rvert+16)$	0	$\beta\cdot(\lvert cert\rvert+16)$	$\beta\cdot(4.19\times10^6)$	$\beta\cdot(4.19\times10^6)$
Ratio: ours/SCMS	0.5	0.5	0.5	0.5	0.86	[0.67, 1[‡	0	[0.67, 1[‡	0.72	0.89

* Assuming that ECDSA verification uses the PCA's public signature key $\mathcal{U}$ as a random (RP) or fixed (FP) point

‡ Ignoring encryption overhead and assuming $|cert| \geqslant 48$, which is close to the minimum for a certificate (32 bytes for representing U_i or V_i, plus 16-bytes for metadata such as the expiration date and vehicle identifier)

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.

to prevent side-channel timming attacks

- PCA: The processing savings come from the fact that each (implicit or explicit) certificate issued takes a single signature instead of two. Inbound and outbound bandwidth are also saved, 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 bandwidth costs are measured 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.). The processing costs are measured in cycles, using the RELIC cryptography library version 0.4.1 [AG18] running on an Intel i5 4570 processor. For completeness, we consider two different settings for ECDSA when measuring the processing costs of the batch verification by vehicles: a standard implementation, in which the verification process takes basically one fixed-point EC multiplication by the generator G and one random-point multiplication by the PCA's signature key $\mathcal{U}$; and an optimized implementation, in which $\mathcal{U}$ is also considered a fixed point. More precisely, RELIC relies on pre-computation for fixed-point EC multiplications, using the fixed comb method with $w = 8$. For the random-point multiplication, RELIC is set to use the Montgomery ladder method, thus providing an isochronous operation. As a result, fixed-point multiplications end up being ≈ 8 times faster than their random-point counterparts. In practice, the adoption of this optimized version is interesting because multiplications by $\mathcal{U}$ are expected to be performed multiple times per batch, so the underlying pre-computation costs can be amortized. Nevertheless, real-world deployments may involve multiple values of $\mathcal{U}$ per batch, e.g., because the RA's policy dictates that different PCAs are contacted so the revocation of one PCA does not invalidate the entire batch. In this latter case, the standard implementation may be preferred over the one that turns $\mathcal{U}$ into a fixed point.

As shown in Table 3, the bandwidth and processing gains of the proposed unified butterfly key expansion process can reach up to 50%, whereas in the worst case it is at

least as efficient as SCMS's original approach. It is interesting to note that those gains are slightly more significant in the implicit certification model, which is the approach suggested for standardization [CAM16].

As a final remark, notice that the certificate verification process at the vehicles is more efficient in the explicit model than in the implicit model when $\mathcal{U}$ is considered a fixed point. This happens because, with pre-computations, the ECDSA verifications performed in the explicit model do not involve any random-point multiplication. In comparison, the verification of implicit certificates requires a (random point) multiplication by the credential V_i; since V_i is unique per pseudonym certificate, there is no advantage in using pre-computation methods for accelerating this operation. Despite these higher processing costs at this point, the adoption of implicit certificates does make sense in the context of vehicular communications for at least two reasons (both of which are highlighet in [Cer13, page17]: (1) it potentially leads to smaller certificates and, hence, to lower bandwidth usage in V2X communications, and (2) during operation, the process of verifying a message's signature can be combined with public key verification for better computational efficiency.

8 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 301198/2017-9, 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+11] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In *Cryptographic Hardware and Embedded Systems – CHES 2011*, pages 124–142, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[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. `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.

[LS08] Kristin E Lauter and Katherine E Stange. The elliptic curve discrete logarithm problem and equivalent hard problems for elliptic divisibility sequences. In *International Workshop on Selected Areas in Cryptography*, pages 309–327. Springer, 2008.

[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, 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, 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, National Institute of Standards and Technology, U.S. Department of Commerce (NIST). 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, 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.

[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.

[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, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.