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]

***Abstract*— With the increasing demand for intelligent transportation systems (ITS), security and privacy requirements are paramount. This led to many proposals aimed at creating a Vehicular Public Key Infrastructure (VPKI) able to address such prerequisites. Among them, the Security Credential Management System (SCMS) is particularly promising, providing data authentication in a privacy-preserving manner and also supporting revocation of misbehaving vehicles. Despite SCMS's appealing design, in this paper we show that its certificate issuing process can be further improved. Namely, one of the main benefits of SCMS is its so-called butterfly key expansion process, which issues arbitrarily large batches of pseudonym certificates by means of a single request. Although this protocol requires the vehicle to provide two separate public/private key pairs to registration authorities, we hereby propose an improved approach that unifies them into a single key. As a result, the processing and bandwidth utilization for certificate provisioning are reduced from 10% to 50% for all entities involved in the protocol. We also show that such performance gains come with no negative impact in terms of security, flexibility or scalability when compared to the original SCMS.**

I. Introduction

In the last decade, the automotive industry has been enhancing 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) [12], [10]. Such applications usually involve the metering of road conditions (e.g., its slope and current traffic) and the vehicles' state (e.g., velocity, position and distance to other cars) [20]. This information can then be shared among vehicles and further relayed to drivers, facilitating timely intervention – either manually or (semi-)automatically – for improving transportation safety and efficiency.

Albeit promising, the large scale deployment of ITS technologies still faces some important challenges, especially security and privacy concerns [22], [8]. 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: although vehicles might still be tracked by traditional means (e.g., via video surveillance), at least the underlying V2X architecture should not facilitate this task.

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 [13]). Among them, one approach of particular interest relies on pseudonym certificates [21]. 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 vehicle must be identified, such as proving that it is authorized to obtain new pseudonym certificates.

There are today many pseudonym-based security solutions for V2X [21]. Examples include the pseudonym scheme with user-controlled anonymity (PUCA) [8] and the Issue First Activate Later (IFAL) scheme [25]. However, since the former was designed to prevent revocation even in case of misbehavior and the latter allows certificate authorities to track vehicles [23], the Security Credential Management System (SCMS) [26], [4] still remains as a more prominent solution for adoption in practice. Actually, SCMS is today one of the leading candidate designs for protecting vehicular communications in the United States [4]. 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. Even though SCMS does not provide formal security proofs, it was designed in such a manner that, unless RA and PCA collude, they are unable to

[1]Escola Politénica, USP, Brazil. [2]LG Electronics, USA. Email: {mjunior,ecominetti,joliveira,mvsilva}@larc.usp.br, harsh.patil@lge.com

link a pseudonym certificate to its owner nor learn whether 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 a misbehaving vehicle, a small piece of information is placed in the Certificate Revocation List (CRL), thereby exposing the culprit and revoking its respective certs.

Although 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 that reduces approximately by half the processing and bandwidth costs of SCMS's certificate provisioning protocol. 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 II summarizes the notation employed along the document. Section III describes with some detail the SCMS protocol, in particular its butterfly key expansion process. Section IV then presents our proposed improvements to the original SCMS, analyzing the resulting security and performance. Finally, section VII concludes the discussion.

II. General Notation

For convenience of the reader, Table I 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 [17] whereas asymmetric encryption (i.e., involving public/private key pairs) is performed with ECIES [11]; hashing is performed with SHA-2 [19]. and the creation and verification of digital signatures uses algorithms such as ECDSA [18].

TABLE I
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 $\mathcal{U}$
S, s	Public and private caterpillar signature keys
E, e	Public and private caterpillar encryption keys
$\hat{S}, \hat{s}$	Public and private cocoon signature keys
$\hat{E}, \hat{e}$	Public and private cocoon encryption keys
X, x	Public and private unified caterpillar keys
$\widehat{X}, \hat{x}$	Public and private unified cocoon keys
β	Number of cocoon keys in certificate batch
σ	Number of certificates valid at any time period
f, f_1, f_2	Pseudorandom functions
$Enc(\mathcal{K}, str)$	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

III. The Security Credential Management System

In this section, we describe SCMS in more detail. Along the discussion, we focus on the description given in [26] rather than in [4]. 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 [26] done in [4].

In SCMS, each device receives two types of certificates: a long-term enrollment certificate, which identifies 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 [7], in which one vehicle poses as a platoon aiming to get some advantage over their peers [16], [1]. 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:

- 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.

These entities play different roles in the two main proce-

dures 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. Due to limited space, we describe only the former procedures in what follows. Also, since all communications between SCMS entities must be performed via a secure channel [4, Sec. 2.2.11.4], we omit this extra layer of security while describing their interactions.

A. 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 1. First (Step 1), the vehicle 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 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). Then (Step 2), the RA employs S 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 [26] differs from the description given in [4], but since the differences are not pertinent to our discussion, we hereby omit their internal details. Pairs of cocoon keys $(\hat{S}_i, \hat{E}_i)$ from different vehicles are then shuffled together by the RA and sent individually to the Pseudonym Certificate Authority (PCA), which generates the corresponding pseudonym certificates (Step 3).

After receiving the cocoon keys from the RA, the PCA can create explicit or implicit certificates [5] (Step 4). In the explicit model, 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. It then inserts U_i into a certificate $cert_i$ containing the required metadata `meta` (e.g., a validity period), 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, generating the PCA's response package. The PCA's response is then digitally signed (step 5) and sent to the RA (step 6), which relays it (in batch) to the requesting vehicle (step 7). As a result of this process, only the corresponding vehicle can decrypt the PCA's responses to learn U_i and compute the corresponding private signature key $\mathrm{u}_i = s + r_i + f_1(i)$ (step 8). To ensure this is the correct key, the vehicle should also perform a final verification $\mathrm{u}_i \cdot G \stackrel{?}{=} \mathrm{U}_i$.

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 $\hat{E}_i$ and signed with the PCA's private key. The vehicle, after decrypting the PCA's response and checking its signature, computes $h_i = Hash(cert_i)$ and sets its own private signature key to $\mathrm{u}_i = h_i \cdot (s + f_1(i)) + sig_i$, whereas

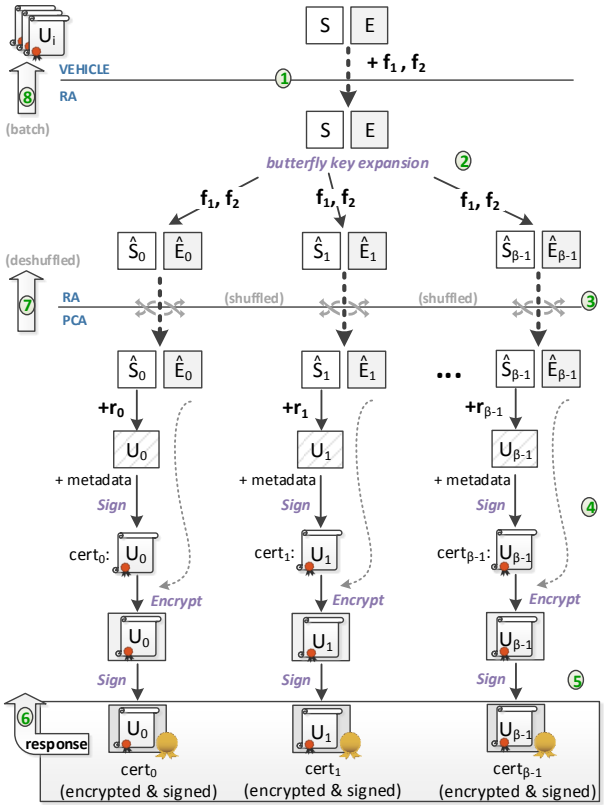

Fig. 1. SCMS's butterfly key expansion and certificate generation – only explicit certificates are illustrated. Numbers in circles indicate the sequence of steps involved in this process.

the corresponding public signature key takes the form $\mathrm{U}_i = \mathrm{u}_i \cdot G$. The validity of the public key U_i can then be implicitly verified by ascertaining that $\mathrm{U}_i = h_i \cdot V_i + \mathcal{U}$, where $\mathcal{U}$ is the PCA's public signature key.

We note that, for both certificate models, the PCA signs the encrypted response using its own private signature key for preventing 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 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 belongs 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.

IV. An improved key expansion process: Unified butterfly keys (UBK)

Albeit quite efficient, in particular from the vehicles' perspective, SCMS's pseudonym certificate provisioning protocol described in Section III-A 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 $0 \leqslant i < \beta$). Furthermore, 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 prevent the RA from manipulating $\hat{E}_i$. This extra signature leads to overheads in multiple places: on the PCA, for its computation and transmission; 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.

Aiming to avoid the need of such extra signature while preserving SCMS's original certificates' format and also improving the overall efficiency of the certificate issuance procedure, we hereby propose a unified butterfly key (UBK) expansion process. In the proposed solution, described in what follows and summarized in Table II, the vehicle's request takes a single key. This leads to better efficiency without loss of security or functionality.

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 the steps followed are analogous to those described in Section III-A. Namely, for implicit certificates, the PCA uses $\widehat{X}_i$ to generate the vehicle's credential $V_i = \widehat{X}_i + r_i \cdot G$, builds the certificate $cert_i = (V_i, \texttt{meta})$, and computes the corresponding signature $sig_i = h_i \cdot r_i + u$, where $h_i = Hash(cert_i)$. In the explicit model, in turn, the device's public signature key is computed directly from $\widehat{X}_i$, by making $\mathrm{U}_i = \widehat{X}_i + r_i \cdot G$; this key is then inserted into the certificate $cert_i$, which is signed by the PCA using u.

Whether implicit or explicit certificates are adopted, $\widehat{X}_i$ is employed by the PCA as the encryption key for its response. In other words, the PCA uses $\widehat{X}_i$ to encrypt $cert_i$ and also any additional data that needs to be provided 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 relays it to the vehicle.

TABLE II

Issuing pseudonym certificates: the original SCMS and the proposed UBK in the implicit and explicit models. Operations saved in the UBK approach are shown in gray background.

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

The vehicle can then compute the corresponding decryption key $\hat{x}_i = x + f(i)$ and 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, whereas the corresponding private key is known only to the vehicle. For implicit certificates, U_i is then computed and verified as usual, using $cert_i$ and the PCA's public signature key to check that $\mathrm{U}_i = \mathrm{u}_i \cdot G$ satisfies $\mathrm{U}_i = h_i \cdot V_i + \mathcal{U}$, where $\mathrm{u} = Hash(cert_i) \cdot (x + f(i)) + sig_i$. For explicit certificates, the vehicle (1) verifies the PCA's signature on $cert_i$, which encloses U_i, and then (2) computes $\mathrm{u}_i = r_i + \hat{x}_i$ using the value of r_i received in the encrypted package, ascertaining that $\mathrm{u}_i \cdot G = \mathrm{U}_i$. 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 V-C).

V. Security analysis

Even though SCMS does not provide formal proofs of security, its underlying goals are basically: the confidentiality of the vehicle's private key u; the confidentiality of the PCA's response (or, more precisely, of the pseudonym certificate thereby enclosed), in particular toward the RA; the integrity of the pseudonym certificates in the PCA's response; and the unlinkability of the pseudonym certificates, as long as PCA and RA do not collude.

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 the process that defines how the caterpillar and cocoon signature keys are handled by PCA and RA. Therefore, the security arguments of SCMS regarding the confidentiality of u, which rely basically on the fact that x remains protected by the elliptic curve discrete logarithm problem (ECDLP, given in Definition V) during the whole execution of the protocol, remain valid. Hence, neither RA nor PCA is able to recover

the signature or decryption private keys derived from it, even if they collude. Certificate unlinkability is also preserved as long as RA and PCA do not collude: the shuffling done by the RA still hides from the PCA any relationship between certificate requests for the same vehicle; meanwhile, the PCA's encrypted response prevents anyone but the owner of the decryption key from learning $cert_i$.

*Definition:*Elliptic Curve Discrete Logarithm Problem (ECDLP) [15]. 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, affecting the confidentiality and/or integrity of issued certificates. 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 such that any manipulation by the RA is detectable by the device when validating the public key U_i, either explicitly or implicitly.

A. 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 strategy 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 Theorem V-A.

Theorem: Security of the unified butterfly key (UBK) 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. ■

B. Security against MitM attacks by RAs (implicit model)

The security result obtained in Section V-A 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 V-B.

Theorem: Security of the unified butterfly key (UBK) 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 [5, Section 3.4], and yet such model ensures the integrity of $cert_i$ in the random oracle model assuming the hardness of the ECDLP [3]. 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 $z = x + f(i)$:

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

Hence, to bypass the vehicle's verification, the RA cannot just choose any z: 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 in this scenario is equivalent to solving the ECDLP for X.

■

C. Security against MitM attacks by RAs (explicit model)

The security arguments for explicit certificates are similar to those presented in Section V-B for the implicit model, as summarized in Theorem V-C.

Theorem: Security of the unified butterfly key (UBK) 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 $(r_i + z) - (x + f(i))$, which is equivalent to solving the ECDLP for the point $(\mathrm{U}_i^* - \widehat{X}_i)$. Actually, since $f(i)$ 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.

■

D. Implementation-related 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 [6]. Hence, if an attacker can somehow interact with a vehicle in such a manner that (1) the vehicle works as an oracle for one process, and then (2) recover the private key thereby employed, then (3) 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. The reason is that 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. Interestingly, this approach follows the same line of thought behind the butterfly key expansion process that is the basis for SCMS:

different signature cocoon keys are generated from the same secret information (the caterpillar key), but this correlation is known only by the vehicle and a system entity (in this case, the RA). Therefore, the proposed modification can be seen as a natural development of the original SCMS protocol.

Finally, as an exercise, it is useful to consider which kind of implementation flaw would be necessary to jeopardize the resulting system's security. We start by noticing that, even if the signature key u_i is somehow compromised, recovering $\hat{x}_i$ as a result of this flaw would only be feasible by the PCA, since it would still be the only entity with knowledge of the r_i or sig_i associated to the compromised U_i. However, the PCA would gain nothing by doing so, because it already knows the plaintext protected with $\hat{x}_i$: after all, the PCA is the one who encrypted that plaintext in the first place. Hence, the only implementation issue that might lead to a useful attack against the UBK process refers to the compromise of the encryption key $\hat{x}_i$. Such attack would require capturing the PCA's response carrying r_i and sig_i, so u_i can be recovered and messages can be forged with it. Once again, however, this is only feasible by an RA or PCA, but not by external entities. The reason is that the PCA-RA and RA-vehicle communications are protected using secure (e.g., TLS-based) channels [4], so RA and PCA are the only entities (besides the vehicle itself) with access to the PCA's response. The RA and the PCA, on the other hand, would not gain much (if anything) by engaging in such attacks, since they could themselves create valid certificates and sign the corresponding messages.

VI. 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 batches is smaller than in SCMS, since the verification of the PCA's signature on the encrypted package is eliminated.
- 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 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 III compares the estimated costs of the proposed procedure with the original SCMS as described in [4], 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., 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 [2], 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 ECDSA 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,fixed-point EC multiplications in RELIC rely on pre-computations, using the fixed comb method with $w = 8$ For random-point multiplications, in turn, 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, at the cost of extra memory usage for storing pre-computed tables. In practice, the adoption of this optimized version is interesting when multiplications by $\mathcal{U}$ are performed multiple times per batch for verifying ECDSA signatures from a same PCA, so the underlying pre-computation costs can be amortized. Nevertheless, in actual deployments such optimized version may not be adopted, for at least two reasons: (1) vehicles may not have enough memory for storing pre-computed tables; and (2) the batch verification may involve multiple values of $\mathcal{U}$, 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, as well as for improved privacy. 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 III, 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. Interestingly, those gains are slightly more significant in the implicit certification model, which is the one suggested for standardization [4].

In practice, whereas such benefits at the vehicles' side are clearly important given their resource-constrained nature, we argue that they are also relevant at the servers' side due to the large scale of the vehicular networking scenario. More precisely, to put these savings in perspective, we can consider the case of the PCA: more than 16 million vehicles are sold per year only in US [24]); since the number of certificates provisioned per vehicle is expected to range from a few thousands [26] to more than 30 thousand [14], avoiding one signature per certificate translates to 16 – 480 billion signatures that do not need to be computed or transmitted per

TABLE III

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 EXPLICIT AND IMPLICIT CERTIFICATES, INCLUDING REQUEST AND RESPONSE.

	Vehicle	$\rightarrow$	RA	$\rightarrow$	PCA	$\rightarrow$	RA	$\rightarrow$	Vehicle (RP)*	Vehicle (FP)*
SCMS expl.	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	$\beta\cdot(\lvert cert\rvert+80)$	$\beta\cdot(5.30\times10^6)$	$\beta\cdot(3.23\times10^6)$
UBK expl.	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	$\beta\cdot(\lvert cert\rvert+48)$	$\beta\cdot(3.75\times10^6)$	$\beta\cdot(2.73\times10^6)$
UBK/SCMS	0.5	0.5	0.5	0.5	0.88	[0.75, 1[‡	–	[0.75, 1[‡	0.71	0.85
SCMS impl.	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	$\beta\cdot(\lvert cert\rvert+48)$	$\beta\cdot(5.74\times10^6)$	$\beta\cdot(4.72\times10^6)$
UBK impl.	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	$\beta\cdot(\lvert cert\rvert+16)$	$\beta\cdot(4.19\times10^6)$	$\beta\cdot(4.19\times10^6)$
UBK/SCMS	0.5	0.5	0.5	0.5	0.86	[0.67, 1[‡	–	[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 linkage values)

year by PCAs in US. Those savings alone are, thus, orders of magnitude larger than the total load of the world's largest PKI, which issues under 10 million certificates every year, run by the US Department of Defense [26]. In particular, if the signatures are generated using a modern Hardware Security Module (HSM), capable of processing 20,000 ECC-based signatures per second [9], the processing time saved at PCAs provisioning that many signatures would be equivalent to 9–277 days of that HSM's continuous operation.

VII. 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 [26] to tens of thousands [14]. In addition, whereas these gains are more noticeable at the vehicles' side, which are exactly the most resource-constrained entities in the system, they are also relevant from the PCA's and RA's perspective given the large number of certificates that need to be handled by the system. The proposed schemes support either implicit or explicit certificates, while still preserving the system's security, flexibility and scalability in both approaches.

Acknowledgements. This work was supported by the Brazilian CNPq (grant 301198/2017-9) and by LG Electronics.

REFERENCES

[1] 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.

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

[3] Daniel Brown, Robert Gallant, and Scott Vanstone. Provably secure implicit certificate schemes. In *Financial Cryptography*, pages 156–165, Berlin, Heidelberg, 2002. Springer.

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

[5] 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`.

[6] Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier. Universal padding schemes for RSA. In *Advances in Cryptology (CRYPTO'02)*, pages 226–241, London, UK, UK, 2002. Springer.

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

[8] 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, 2014.

[9] Gemalto. SafeNet Luna Network HSM - product brief. `https://safenet.gemalto.com/`, 2018.

[10] 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, NHTSA, 2014.

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

[12] 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.

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

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

[15] Kristin E Lauter and Katherine E Stange. The elliptic curve discrete logarithm problem and equivalent hard problems for elliptic divisibility sequences. In *Selected Areas in Cryptography (SAC'08)*, pages 309–327. Springer, 2008.

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

[17] NIST. *FIPS 197 – Advanced Encryption Standard (AES)*. National Institute of Standards and Technology, November 2001.

[18] NIST. *FIPS 186-4 – Digital Signature Standard (DSS)*. National Institute of Standards and Technology, July 2013.

[19] NIST. *FIPS 180-4 – Secure Hash Standard (SHS)*. National Institute of Standards and Technology, August 2015.

[20] 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.

[21] 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.

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

[23] M. Simplicio, E. Cominetti, H. Kupwade Patil, J. Ricardini, L. Ferraz, and M. Silva. A privacy-preserving method for temporarily linking/revoking pseudonym certificates in vanets. In *17th IEEE Int. Conf. On Trust, Security And Privacy In Computing And Communications (TrustCom'18)*, 2018. See also `https://eprint.iacr.org/2018/185`.

[24] Statista. U.S. car sales from 1951 to 2017 (in units). `www.statista.com/statistics/199974/us-car-sales-since-1951/`, 2018.

[25] E. Verheul. Activate later certificates for V2X: Combining ITS efficiency with privacy. Cryptology ePrint Archive 2016/1158, 2016.

[26] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn. A security credential management system for V2V communications. In *IEEE Vehicular Networking Conference (VNC'13)*, pages 1–8, 2013.