Privacy-preserving linkage/revocation of VANET certificates without Linkage Authorities

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

[1] Escola Politécnica, Universidade de São Paulo, Brazil,
{mjunior,ecominetti,lferraz,mvsilva}@larc.usp.br
[2] LG Electronics, USA,
{harsh.patil,jefferson1.ricardini@lge.com}@lge.com

Abstract. Vehicular communication (V2X) technologies are expected to be common in the future, providing better transportation safety and efficiency. However, their large-scale deployment requires addressing some challenges. In particular, to prevent abuse by drivers and by the system itself, V2X architectures must: (1) ensure the authenticity of messages, which is usually accomplished by means of digital certification; and (2) preserve the privacy of honest users, so owners of non-revoked certificates cannot be easily identified or tracked by eavesdroppers. A promising solution for managing V2X-oriented certificates in an efficient manner is the Security Credential Management System (SCMS), which is among the main candidates for standardization in the United States. In this paper, aiming to enhance and address issues in the SCMS architecture, we provide three main contributions. First, we describe and fix two birthday attacks against SCMS's certificate revocation process, thus preventing the system's security degradation with the number of issued and revoked certificates. In addition, we describe a mechanism for improving the flexibility of revocation, allowing certificates and their owner's privacy to be temporarily revoked in an efficient manner; this functionality is useful, for example, in case of vehicle theft or kidnapping. Finally, we propose a method that simplifies SCMS's system architecture, removing the need for the so-called Linkage Authorities (LAs); this not only results in cost reductions for SCMS's implementation, but also improves its security and privacy due to the removal of one potential point of failure/collusion.

Keywords: No keywords given.

1 Introduction

The past decade has witnessed a surge in digital technologies embedded in physical objects, leading to what today we know as the Internet of Things (IoT). This trend has also reached the automotive industry, which has shown a growing interest in exploring interaction models such as Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I) and Vehicle-to-Pedestrian (V2P), collectively referred to as Vehicle-to-Everything (V2X) communications [1].

V2X enables several applications aimed at improving transportation safety, efficiency, and human to machine interaction [2]. For example, information classified as Basic Safety Messages (BSMs) – which include velocity, direction and brake status – can help drivers to keep a safe distance from other vehicles while maintaining a suitable speed. Actually, onboard systems can evaluate such information and, if an accident appears to be imminent, the vehicle itself can provide (semi-)automatic responses to prevent it in a timely manner.

These prospects are among the motivations behind a recent publication by the United States Department of Transportation (USDOT) [3] mandating that vehicles should be capable of exchanging BSMs with each other.

Albeit promising, the large scale deployment of V2X requires addressing some challenges, especially security and privacy concerns [4]. More precisely, V2X architectures are expected to: (1) ensure the legitimacy of messages exchanged between vehicles, banning users in case of misbehavior; and (2) preserve the anonymity of honest users, so their movements cannot be easily tracked by any entity. One common approach for fulfilling these requirements is to create a Vehicular Public-Key Infrastructure (VPKI) [4][5]. In this case, the system's security involves issuing digital certificates to vehicles, so they can sign their own messages. Such certificates can, however, be revoked if the system detects some misbehavior, such as the transmission of invalid messages (e.g., due to a malfunction or for malicious purposes). Vehicles are then expected to verify the authenticity of received messages, acting upon them only if signed by a non-revoked peer.

Ensuring the vehicles' privacy, in turn, requires the vehicles' certificates to be devoid of any information that identifies their owners. Otherwise, by monitoring when and where messages signed by vehicles are broadcast, an eavesdropper can track a given target. One promising approach to alleviate such issues is to rely on pseudonym certificates [6], in which random-like strings play the role of identifiers. By signing different messages with distinct pseudonym certificates, those messages cannot be linked to the same vehicle, thus preserving the sender's privacy. Pseudonym certificates are usually short-lived (e.g., valid for one week), which contributes to their owner's privacy and facilitates revocation. The usage of traditional, 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. The frequent renewal of pseudonym certificates should be avoided, though, because vehicles are not expected to constantly have access to a reliable network connection. Hence, vehicles are usually provisioned with batches of pseudonym certificates covering current and future time periods, thus enabling their operation for a long time (e.g., years) [7][8].

Among the many pseudonym-based security solutions for V2X (see [6] for a survey), one of the most prominent is the Security Credential Management System (SCMS) [7][9]. Indeed, this solution is currently considered a leading candidate for protecting vehicular communications in the United States [9]. One of the main merits of SCMS is that vehicles can obtain arbitrarily large batches of (short-lived) pseudonym certificates from a Pseudonym Certificate Authority (PCA) by means of a single, small-sized request sent to a Registration Authority (RA). This process is such that, unless RA and PCA collude, they are unable to link a pseudonym certificate to its owner, nor to learn whether two pseudonym certificates belong to the same vehicle. In case of abuse, however, the misbehaving vehicle's privacy is annulled, while its pseudonym certificates are revoked altogether by placing a small piece of information in a Certificate Revocation List (CRL). To accomplish this, SCMS requires RA and PCA to collaborate with at least two Linkage Authorities (LAs), which supply the information inserted into pseudonym certificates for enabling such an efficient revocation procedure.

Motivated by this growing practical interest in SCMS, some recent studies unveiled opportunities for enhancing its design, proposing novel or alternative procedures for tackling identified issues [8][10][11]. In this article, we contribute to this research effort by presenting three independent improvements to SCMS's certificate revocation and linkage approach, enhancing its security and flexibility while reducing the underlying architecture's complexity. Specifically, (1) while the original SCMS focused just on the permanent revocation of devices, we describe an efficient method that enables vehicle revocation and linkage for a limited time. Such capability is useful, for example, when vehicles need to be temporarily suspended, or when aiding in investigations by law enforcement authorities.

In addition, (2) we show that SCMS's revocation procedure is prone to attacks built upon the birthday paradox to degrade the VPKI's security over time. We then propose an alternative method that addresses this issue with minimal overhead. Finally, (3) we describe a mechanism that simplifies SCMS's overall architecture, removing the need for LAs. Namely, we modify SCMS's design in such a manner that the LAs' roles can be securely played by PCA and RA, thus avoiding the additional point of failure/collusion represented by LAs in the original SCMS. This article extends and consolidates our previous study, published in [12], in which building such an LA-free solution was left as a topic for future work.

The remainder of this paper is organized as follows. Section 2 lists the notation and main symbols employed in the document. Section 3 discusses some related works, giving a broad view of the state of the art on V2X security and privacy. Section 4 describes SCMS, focusing on its key linkage and revocation process. Section 5 presents our first contribution, an efficient method for enabling the temporary revocation and linkage of certificates. Section 6 shows how to build and fix birthday attacks against SCMS's key linkage, which is our second contribution. Our third contribution is described and analyzed in Section 7, in which we show how to build an LA-free SCMS. Section 8 summarizes and concludes the article.

2 General notation

For convenience, we list in Table 1 the symbols and general notation adopted in this paper.

When cryptographic algorithms are mentioned, we assume standardized schemes are adopted, providing a security level of at least 128 bits. In particular, the following algorithms are considered safe for use in modern systems: symmetric encryption (i.e., using secret keys) is performed with the AES block cipher [13] whereas asymmetric encryption (i.e., using public/private key pairs) is done with ECIES [14]; the hashing algorithm may be either SHA-2 [15] or SHA-3 [16]; and digital signatures are generated and verified with ECDSA [17] or EdDSA [18][19].

3 V2X security and privacy: state of the art

Following the emergence of V2X, many proposals have appeared in the literature aiming to fulfill its security and privacy requirements. Notably, approaches based on pseudonym certificates seem to be promising, and are being considered in standardization efforts [20][21]. Existing schemes rely on a variety of security primitives, including asymmetric, identity-based, and symmetric cryptography, as well as on group signatures (for a survey, see [6]). In what follows we discuss some recent works, focusing on how they handle a vehicle's revocation, giving an overview of the state of the art.

The Pseudonym scheme with User-Controlled Anonymity (PUCA) [22] was designed to ensure that the end users' privacy is preserved even when multiple system entities collude. PUCA assumes that each vehicle is equipped with a trusted module, which manages long-term enrollment certificates, cryptographic keys and pseudonyms. This module is also responsible for handling revocation: if a vehicle misbehaves, the system broadcasts an order of self-revocation (OSR) to the corresponding pseudonym; the trusted module responsible for that pseudonym is then expected to halt its operation. With this approach, the users' privacy is preserved even in case of misbehavior, since there is no need to identify the real users behind revoked pseudonyms. One potential issue, however, is that malicious users might somehow discard OSR messages addressed to them or tamper with the trusted module, thus avoiding revocation. In addition, PUCA was designed to prevent pseudonym certificates from being linked together, which might be reasonable if V2X itself

Table 1: General notation and symbols

Symbol	Meaning
G	The generator point for an elliptic curve group $\mathbb{G}$
r	A random value
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 associated to U and $\mathcal{U}$ (respectively)
S, s	Public and private caterpillar keys for signature
E, e	Public and private caterpillar keys for encryption
$\hat{S}, \hat{s}$	Public and private cocoon keys for signature
$\hat{E}, \hat{e}$	Public and private cocoon keys for encryption
β	Number of cocoon keys in a batch of certificates
la_id	ID of a Linkage Authority (LA)
ℓ	Number of LAs (typically two)
$\mathtt{ls}, \mathtt{lh}$	Linkage seed and linkage hook, respectively
$\mathtt{lv}, \mathtt{plv}$	Linkage value and pre-linkage value, respectively
τ	Number of time periods covered by a batch of certificates
t	Index of a time period among the τ possible
σ	Number of certificates valid in each time period
c	Index of a certificate among the σ possible
υ	Number of different key usages in a time period
f_1, f_2	Pseudorandom functions
$Enc(\mathcal{K}, str)$	Encryption of bitstring str with key $\mathcal{K}$
$Hash(str)$	Hash of bitstring str
$str_1 \parallel str_2$	Concatenation of bitstrings str_1 and str_2

did not lead to new threats. Unfortunately, however, this is not the case because malicious users can abuse the V2X capabilities to deliberately cause accidents or facilitate robbery (e.g., inducing vehicles to take a detour or reduce their speed by informing a fake road accident). This is the reason why most V2X solutions argue in favor of revocable privacy as a requirement (e.g., [7][8][11]).

One interesting solution aimed at certificate revocation in the V2X scenario is the Issue First Activate Later (IFAL) scheme [23]. IFAL provides mechanisms based on “activation codes”, bitstrings that are required for the computation of any pseudonym certificate’s private key. As long as a revoked vehicle does not receive the activation codes for pseudonym certificates obtained prior to revocation, it is prevented from using those certificates even if they are still valid. Non-revoked vehicles can then periodically request the activation codes for their own certificates, whereas requests from revoked vehicles are ignored. This results in a reduction of the CRLs’ sizes, since entries corresponding to that vehicle (or to its pseudonyms) do not need to remain in the CRL. Two similar-purpose solutions, Binary Hash Tree based Certificate Access Management (BCAM) [8] and Activation Codes for Pseudonym Certificates (ACPC) [11], achieve an equivalent reduction on the size of CRLs while introducing a more efficient process for distributing activation codes.

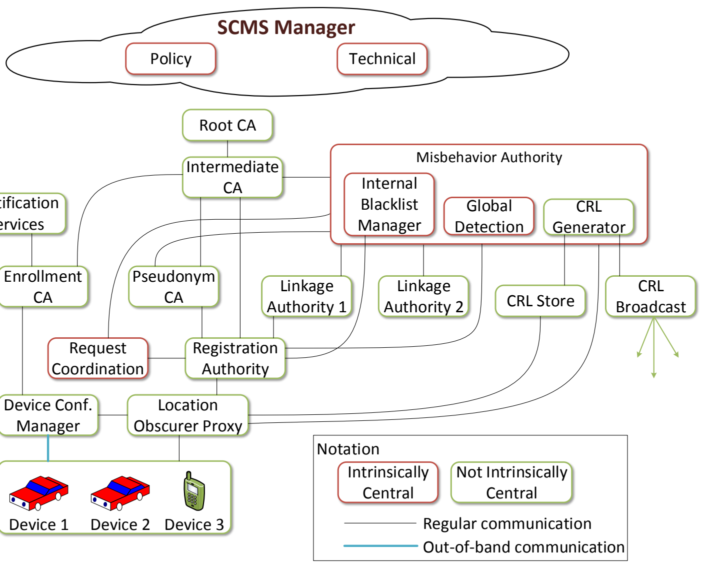

Figure 1: Overview of SCMS's architecture. Source: [7].

Namely, activation codes can be computed from a small piece of information broadcast by a Certificate Access Manager (CAM), so vehicles are not required to explicitly request them. ACPC is advantageous over BCAM, though, in particular: better security, due to the fact that a dishonest CAM cannot track vehicles even if it colludes with other system entities (unlike BCAM); and a considerably smaller processing and bandwidth usage. Since IFAL, BCAM and ACPC aim to reduce the number of identifiers placed in CRLs, without affecting how these identifiers are constructed or organized, they can actually be seen as complementary to the mechanisms hereby presented.

Finally, the Security Credential Management System (SCMS), originally proposed in [7] and later extended in [9], is one of the main proposals in the literature for dealing with revocable privacy while preventing non-colluding system entities from tracking devices. These security properties are expected to hold in the so-called "honest-but-curious" threat model: even though the system's entities follow the correct protocols when issuing and revoking pseudonym certificates, they might engage in passive attacks, trying to use the information acquired during the protocols' execution to their advantage (e.g., to track vehicles) [4]. Since SCMS is one of the leading candidates for protecting V2X security in the US [7][9], and the proposed improvements are built upon its certificate linkage and revocation procedure, we analyze the latter more closely in Section 4.

4 The Security Credential Management System

In SCMS, each vehicle receives two types of certificates: an enrollment certificate, which has a long expiration time (e.g., years) and identifies legitimate devices; and multiple pseudonym certificates, each valid for a short time period (e.g., a few days), so $\sigma \geqslant 1$ certificates of this type are valid simultaneously. For privacy reasons, vehicles are expected to frequently change the pseudonym certificate employed for signing messages, thus avoiding tracking by eavesdroppers. However, the value of σ should be limited to avoid "sybil-like" attacks [24], in which one vehicle pretends to be a platoon by signing multiple messages with different pseudonyms [25][26]. For example, if traffic lights give higher priority to congested roads, such a fake platoon might receive preferential treatment while driving on lightly loaded roads.

SCMS's design is such that batches of pseudonym certificates can be efficiently distributed to vehicles, as well as revoked in case of misbehavior by their owners. The basic architecture that gives support to these capabilities involves the following entities (see Figure 1 for the complete architecture and [7] for the description of all of its elements):

- Registration Authority (RA): handles requests for batches of pseudonym certificates. It acts as a proxy, turning a vehicle's request carrying a single public key into multiple individual requests for the PCA, with distinct keys. Requests from different vehicles are shuffled together, thus preventing the PCA from linking groups of certificates to the same requester. In addition, when revocation is required, the RA helps identifying the corresponding certificates.
- Pseudonym Certificate Authority (PCA): issues batches of pseudonym certificates to the vehicles in response to requests by RAs. The PCA randomizes the public keys placed inside the certificates and encrypts its responses, thus ensuring that RAs do not learn the contents of those certificates (and, thus, cannot easily track the corresponding vehicles). During the revocation process, it also collaborates with the RA for identifying which request generated the pseudonym certificate under scrutiny.
- Misbehavior Authority (MA): monitors the system for signs of misbehavior. When a misbehaving vehicle is identified, its pseudonym certificates are linked together and revoked, along with the corresponding enrollment certificate. This is accomplished by placing that vehicle's information into a Certificate Revocation List (CRL). As a result, messages signed by that vehicle are not considered valid anymore, and it becomes unable to request new pseudonym certificates.
- Linkage Authority (LA): generates linkage values that are inserted in the pseudonym certificates, as well as placed in the CRL for efficiently revoking them.

Different organizations should run those authorities, aiming to reduce the probability of collusion [27]. Also, like SCMS, we hereby assume a "dishonest-if-allowed" security model [11]. This means that authorities may engage not only in passive attacks, but also actively run protocols differently than specified, as long as: (1) some advantage is gained as a result (e.g., the ability to track vehicles), and (2) such misbehavior goes undetected with high probability. Nonetheless, authorities are not expected to abuse their own capabilities (e.g., RAs and PCAs would not request or issue certificates for unauthorized vehicles). In this article, we specifically consider the following attacks under this security model: passive attacks by RAs, PCAs, or third parties; RAs taking advantage of their position as proxy to modify or replay messages; and, although SCMS's design is such that the system's privacy cannot resist a collusion between RAs and PCAs or between two LAs [7], we aim to prevent any privacy degradation due to an RA-LA collusion. We note that this assumption is stronger than the "honest-but-curious" security model found in some V2X systems [4], in which the protocols cannot be subverted.

Such entities participate in SCMS's two main procedures: the "butterfly key expansion", by means of which pseudonym certificates are issued to vehicles; and "key linkage", which allows the revocation of those certificates in case of misbehavior. For completeness, we describe both procedures in what follows, although our proposed improvements are targeted specifically at SCMS's key linkage.

4.1 Butterfly key expansion

SCMS's butterfly key expansion process allows vehicles to obtain arbitrarily large batches of (short-lived) pseudonym certificates by means of a single, small-sized request. It involves

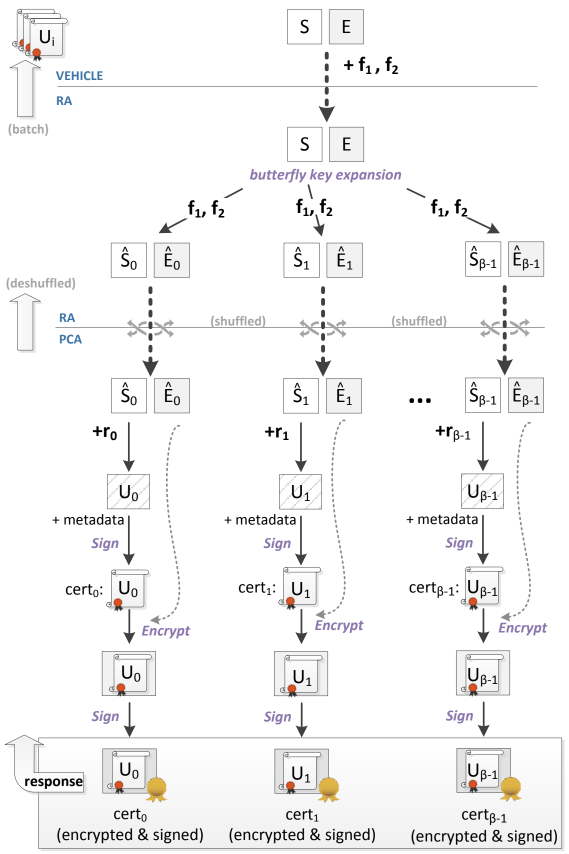

Figure 2: SCMS's butterfly key expansion and certificate generation.

the following steps, which are illustrated in Fig. 2. First, the vehicle generates two pairs of *caterpillar* private/public keys, $(s, S = s \cdot G)$ and $(e, E = e \cdot G)$, for randomly picked s and e. In addition, the vehicle also instantiates two pseudorandom functions (PRF), f_1 and f_2, which are sent to the RA together with the public caterpillar keys S and E.

The RA, in turn, uses S and f_1 to generate 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, E and f_2 are employed in the computation of β public cocoon encryption keys $\hat{E}_i = E + f_2(i) \cdot G$. Pairs of cocoon keys $(\hat{S}_i, \hat{E}_i)$ corresponding to different vehicles are then shuffled together by the RA before being individually sent to the PCA.

Upon reception of a $(\hat{S}_i, \hat{E}_i)$ pair from the RA, the PCA creates the vehicle's pseudonym certificates. Assuming an explicit certification model (as opposed to implicit certificates [28]), the vehicle's public signature key is computed as $\mathrm{U}_i = \hat{S}_i + r_i \cdot G$, for a random value r_i. The resulting U_i is then inserted into a certificate $cert_i$ together with the required metadata `meta` (e.g., the corresponding validity period and linkage values, as described later in Section 4.2). Finally, the PCA digitally signs $cert_i = (\mathrm{U}_i, \mathtt{meta})$ with its own private key u, uses $\hat{E}_i$ to encrypt both the signed certificate and the value of r_i, and once again signs the result before relaying it to the RA. As a result, only the requesting vehicle can decrypt the PCA's response with the private decryption key $e + f_2(i)$. By doing so, the vehicle learns the public signature key U_i, and then computes its corresponding private signature key $\mathrm{u}_i = s + r_i + f_1(i)$.

This process preserves the vehicles' privacy as long as there is no collusion between RA and PCA. After all, by shuffling the public cocoon keys from different vehicles together, the RA prevents the PCA from linking groups of keys to the same device. Unlinkability of public keys towards the RA, in turn, is ensured because RAs never learn the value of $cert_i$ from the PCA's encrypted response. Finally, by signing its response the PCA prevents Man-in-the-Middle (MitM) attacks by the RA, which could otherwise replace the vehicle's cocoon encryption key $\hat{E}_i$ with an arbitrary key $\hat{E}_i^*$, decrypt the PCA's response, and then reencrypt the (now known) certificate $cert_i$ with the correct $\hat{E}_i$ [10].

4.2 Key linkage

The revocation process in SCMS avoids creating large CRLs: by placing a small piece of information in a CRL, SCMS allows multiple certificates from a vehicle to be linked together. This is accomplished by including linkage values ($\texttt{lv}$) as part of the pseudonym certificates' metadata, as described in what follows.

Suppose the RA needs to create a batch of pseudonym certificates covering τ time periods, with σ pseudonym certificates valid per period, so the batch size is $\beta = \tau \cdot \sigma$. In this case, the RA chooses $\ell \geqslant 2$ LAs, and requests from each of them β pre-linkage values $\texttt{plv}_i(t, c)$, where $0 \leqslant t < \tau$ and $0 \leqslant c < \sigma$.

In response to the RA's request, LA$_i$ picks a random 128-bit linkage seed $\texttt{ls}_i(0)$. Then, it iteratively computes a τ-long hash chain [29] of the form $\texttt{ls}_i(t) = \mathit{Hash}(la\_id_i \parallel \texttt{ls}_i(t-1))$, where la_id_i is LA$_i$'s identifier, $\parallel$ denotes concatenation, and $1 \leqslant t < \tau$. Each linkage seed $\texttt{ls}_i(t)$ obtained in this manner is then employed in the computation of σ pre-linkage values $\texttt{plv}_i(t, c) = \mathit{Enc}(\texttt{ls}_i(t), la\_id_i \parallel c)$. Finally, every $\texttt{plv}_i(t, c)$ is truncated to a suitable length, individually encrypted and authenticated[1] using a key shared between PCA and LA$_i$, and sent to the RA.

After receiving the LAs' responses, the RA simply includes this (encrypted) information in the pseudonym certificate request sent to the PCA, so $\texttt{plv}_i(t, c)$ (for $1 \leqslant i \leqslant \ell$) accompanies the c-th request for a pseudonym certificate that should be valid at time period t. The PCA, after decrypting the pre-linkage values and verifying their authenticity, computes the linkage value $\texttt{lv}(t, c)$ by XORing those pre-linkage values together. In the usual case, which consists of two LAs, the linkage value for the c-th certificate and time period t is then computed as $\texttt{lv}(t, c) = \texttt{plv}_1(t, c) \oplus \texttt{plv}_2(t, c)$.

With this approach, whenever the MA notices that a pseudonym certificate was involved in some malicious event, certificates from the same owner can be revoked altogether. This requires the collaboration of the PCA, RA, and LAs, as follows. First, the PCA associates the $\texttt{lv}$ informed by the MA to the original pseudonym certificate request from the RA. The PCA then sends this information, as well as the corresponding pre-linkage values $\texttt{plv}_i(t, c)$, to the RA. Subsequently, the RA does the following: (1) identifies the vehicle behind the original request, placing its enrollment certificate in a blacklist to prevent it from obtaining new pseudonym certificates; and (2) asks LA$_i$ to identify the linkage seed $\texttt{ls}_i(0)$ from which $\texttt{plv}_i(t, c)$ was computed. Each LA$_i$ responds with $\texttt{ls}_i(t_s)$, so the revocation (and, consequently, privacy loss) starts being valid at time period t_s. For example, t_s might be the current time period, or the time period when the misbehavior was first detected. The linkage seeds $\texttt{ls}_i(t_s)$ provided by the LAs are then placed in a public CRL. This allows any entity to compute $\texttt{lv}(t, c)$ for time periods $t \geqslant t_s$, identifying which certificates correspond to a CRL entry. Such mechanism ensures forward privacy: the misbehaving vehicle's certificates for *current* and *future* time periods are revoked, and messages signed with the corresponding keys can be traced back to that vehicle; messages signed in *past* time periods cannot be linked, though, preserving the user's privacy prior to the detection of the malicious activity.

[1] Authentication and freshness are not explicitly mentioned in [7][9]. However, they are important properties to prevent a dishonest RA from delivering a forged or reused $\texttt{plv}$ without contacting LA$_i$. This

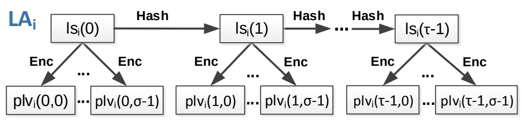

Figure 3: SCMS's key linkage tree: LA$_i$ generates the linkage seeds (`ls`) and pre-linkage values (`plv`) employed for certificate revocation/linkage.

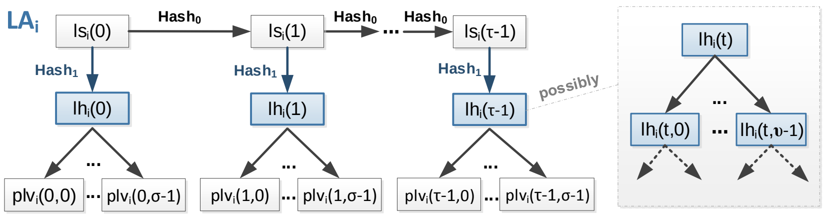

Figure 4: Linkage tree with linkage hooks.

4.2.1 Processing costs

In terms of complexity, SCMS's revocation process is such that, if the VPKI involves ℓ LAs, a total of ℓ pre-linkage values need to be inserted in to the CRL for each revoked device. Hence, the CRL size grows linearly with the number of *revoked devices*, not with the number of *revoked certificates*. Such mechanism is useful not only for saving bandwidth, but also because the larger the number of entries in a CRL, the higher the processing overheads for checking a certificate's revocation status. More precisely, for each CRL entry published at time period t_s, verifying whether it covers a given certificate involves the computation of two components:

a) $\mathtt{ls}_i(t_c)$: it takes $\ell \cdot (t_c - t_s)$ hashes to compute $\mathtt{ls}_i(t_c)$ from $\mathtt{ls}_i(t_s)$, where $1 \leqslant i \leqslant \ell$ and t_c is the time period in which the verification is performed. This cost may be reduced by means of pre-computation, i.e., if the vehicle always keeps the updated version of the linkage seeds, $\mathtt{ls}_i(t_c)$, besides the original ones provided in the CRL. Nevertheless, to cope with the lack of a system-wide time synchronization [23], devices may actually need to keep a slightly older linkage seed in memory; for example, by keeping $\mathtt{ls}_i(t_c - \epsilon)$ for a small ϵ, it is possible to compute $\mathtt{ls}_i(t_c)$ with only ϵ hashes.

b) $\mathtt{plv}_i(t_c, c)$: $\ell \cdot \sigma$ encryptions are required to compute $\mathtt{plv}_i(t_c, c)$ from $\mathtt{ls}_i(t_c)$, since the certificate under analysis may be any out of σ that are valid in the current time period With enough memory, however, the latency of this process can be reduced to ℓ via the pre-computation of a look-up table with all σ possible entries for each $\mathtt{ls}_i(t_c)$ in the CRL.

5 A more flexible revocation/linkage procedure: linkage hooks

One shortcoming of SCMS is that it only allows the permanent revocation of users, via the disclosure of the linkage seed for the current time period. This can be seen in Fig. 3, which shows a graphical representation of the dependencies between linkage seeds and pre-linkage values in the linkage tree: as indicated by the directed arrows, the disclosure of linkage seed $\mathtt{ls}_i(t_s)$ allows anyone to compute the pre-linkage values associated to it, $\mathtt{plv}_i(t_s, \cdot)$, as well as linkage seeds for subsequent time periods and their corresponding pre-linkage values.

misbehavior would allow the RA to track vehicles, as further discussed in Section 7.1.

Even though permanent revocation is indeed a critical use case for key linkage in V2X communications, in some situations it is also important that a small set of messages exchanged among vehicles can be traced to their origin. For example, when handling traffic accidents, the messages sent by the vehicles involved in the event may be useful for law enforcement authorities, so they can understand its causes (and, possibly, identify culprits) [25]. Similarly, a hijacked car might have its privacy temporarily suspended. This would allow nearby vehicles and roadside units to identify all messages coming from it and, thus, track its movements. In the original SCMS, this could be accomplished if the identifiers for all certificates belonging to that time period are placed in a CRL. For improved performance, however, it would be interesting to enable this temporary tracking by means of a single CRL-like entry, similarly to what happens with permanent revocations.

Aiming to enable the temporary revocation of vehicles, while still maintaining SCMS's overall approach, we propose a slight modification to the construction of linkage trees. In the proposed scheme, illustrated in Fig. 4, we create one extra layer to the linkage tree: a *linkage hook* $\mathtt{lh}_i(t)$ is inserted between any linkage seed $\mathtt{ls}_i(t)$ and its corresponding pre-linkage values $\mathtt{plv}_i(t,\cdot)$. This is accomplished by applying a hash function with different suffixes attached to their inputs when deriving linkage seed $\mathtt{ls}_i(t+1)$ and linkage hook $\mathtt{lh}_i(t)$ from the same $\mathtt{ls}_i(t)$ (e.g., a '0' suffix for linkage seeds, and a '1' suffix for linkage hooks). As a result, the disclosure of $\mathtt{lh}_i(t_s)$ allows the recovery of every $\mathtt{plv}_i(t_s,\cdot)$, but not of any $\mathtt{plv}_i(t,\cdot)$ for $t \neq t_s$. This simple modification is, thus, enough to grant the VPKI the ability to link and/or revoke all certificates from a given time period.

In terms of performance, the added flexibility introduced by linkage hooks incurs only a small overhead when compared to the original SCMS scheme. Namely, in any time period, a single extra hash function call is required for verifying whether a device's certificates were permanently revoked, for computing the corresponding linkage hook. Checking whether a certificate was temporarily revoked with a linkage hook, on the other hand, is as costly as verifying if it was (permanently) revoked in SCMS.

5.1 Possible extensions

As a side note, it is worth mentioning that the same concept can be further extended to address other use cases, such as a scenario in which only a part of the certificates from a given time period need to be linked/revoked. For example, suppose that the system's certificates must have υ different purposes, so they display distinct "key usage" fields (like in regular X.509 certificates [30]) even though they share the same validity period. This feature might be the useful, e.g., for protecting the identity of official vehicles: whenever they do not need (or want) to be identified, they could use their regular pseudonym certificates, identical to those issued to other vehicles; however, when they need to prove their status as official vehicles (e.g., aiming to get traffic priority), they would sign their messages with special-purpose certificates.

In this scenario, one possible approach for allowing the independent revocation/linkage of such different-purpose certificates is to create distinct linkage trees, one for each key usage. Then, the certificates sharing the same purpose could be revoked altogether as usual, by inserting ℓ linkage hooks (for temporary revocation) or seeds (for a permanent revocation) in a CRL. However, revoking all certificates belonging to a vehicle would lead to υ times more data placed in CRLs. Conversely, a more efficient revocation can be obtained by adding one extra level to the linkage tree, with $\mathtt{lh}_i(t)$ linking the multiple $\mathtt{lh}_i(t,0)\ldots\mathtt{lh}_i(t,\upsilon-1)$. Hence, if all certificates from a given time period need to be linked/revoked, then $\mathtt{lh}_i(t)$ would be disclosed as usual; conversely, if only certificates of a certain type needed to be linked/revoked, then the disclosure of the corresponding linkage hook would suffice.

6 Building (and preventing) birthday attacks against SCMS's key linkage process

One issue with SCMS's key linkage and revocation procedure is that it is prone to attacks built upon the birthday paradox to degrade the VPKI's security over time, allowing the recovery of linkage seeds that have not been placed in CRLs. This issue appears during the computation of linkage seeds and of pre-linkage values derived from them. In this section, we describe both issues in detail and show how they can be fixed altogether by modifying the linkage tree's internal structure.

6.1 Birthday attacks on pre-linkage values

In SCMS, multiple plvs are computed by encrypting the same plaintext, under different k-bit keys. Namely, for all users, the c-th pre-linkage value valid on a given time period t, $\texttt{plv}_i(t, c)$ is computed by LA_i as $Enc(\texttt{ls}_i(t), la\_id_i \parallel c)$, using the linkage seed $\texttt{ls}_i(t)$ as encryption key.

This procedure allows the construction of a key recovery attack typical of a multi-key setting [31][32], as follows. First, the attacker picks 2^n distinct keys $\texttt{ls}_i^j$, where $0 \leqslant j < 2^n$. Then, the attacker performs 2^n encryptions to build a table of the form $\{\texttt{plv}_i^j, \texttt{ls}_i^j\}$, where $\texttt{plv}_i^j = Enc(\texttt{ls}_i^j, la\_id_i \parallel c)$ for a target la_id_i and a fixed $0 \leqslant c < \sigma$. According to the birthday paradox, if the attacker can gather a total of 2^m pre-linkage values computed by LA_i for the same index c, at least one of those pre-linkage values will match a $\texttt{plv}_i^j$ in the attacker's table with a high probability as long as $m + n \geqslant k$ [31]. Except in the very unlikely case of equivalent keys, whenever there is a match for $\{\texttt{plv}_i^j, \texttt{ls}_i^j\}$ it is safe to assume that $\texttt{ls}_i^j$ corresponds to the linkage seed employed for the computation of that pre-linkage value.

Since the 2^m pre-linkage values employed in the attack can refer to different vehicles and time periods, as long as they receive the same index in that time period and come from the same LA_i, the security of the system for a given choice of k degrades as time passes and LA_i serves more devices. To give some concrete numbers, suppose that 1 million pre-linkage values equally indexed by a given LA are disclosed in one year of the VPKI's operation, meaning that $m \approx 20$. That would be the case, for example, if 20,000 vehicles receive batches of pseudonym certificates in the first week of the year, meaning that pre-linkage values for ≈ 50 weeks become available for the attack. In that case, even if that LA adopts a modern security level of $k = 128$ bits, in the span of one year the system's security against such collision attacks drops to $k - m = 108$ bits. Albeit still high enough to prevent most practical attacks, this security level is below the minimum currently recommended by NIST, of 112 bits [33]. In practice, this may end up limiting the lifespan of LAs for a given security threshold, in particular because the recovery of a given $\texttt{ls}_i(t_s)$ allows the computation of any subsequent $\texttt{ls}_i(t)$ for $t > t_s$. Therefore, the effects of such key-recovery could be quite serious to the affected vehicles' privacy.

Fortunately, SCMS is not completely defenseless against this attack, for at least two reasons. The first is that, by design, only the PCA has access to the raw pre-linkage values not included in CRLs, whereas the device's certificates contain only linkage values (i.e., the XOR of two or more pre-linkage values). Hence, even though the PCA is able to perform the aforementioned attacks, external entities are in principle prevented from doing so. The second is that the cipher's output is actually truncated for the computation of $\texttt{plv}_i(t, c)$ (namely, in [7], the authors suggest using the 8 most significant bytes of AES). This should produce many partial matches on the attacker's table, leading to multiple candidates for the correct linkage seed $\texttt{ls}_i(t)$. Nevertheless, these candidates could still be filtered with a certain probability if the attacker has access to additional pre-linkage values related to the same linkage seed. For example, from $\texttt{plv}_i(t, c)$ and $\texttt{plv}_i(t + 1, c)$, the attacker obtains, respectively, one set of candidates for $\mathtt{ls}_i(t)$ and one for $\mathtt{ls}_i(t+1)$; incorrect candidates can then be filtered out if they do not satisfy $\mathtt{ls}_i(t+1) = \mathit{Hash}(\mathit{la\_id}_i \parallel \mathtt{ls}_i(t))$. Alternatively, if the pre-linkage values obtained are $\mathtt{plv}_i(t,c)$ and $\mathtt{plv}_i(t,c')$, with $c' \neq c$, two tables can be built: one of the form $\{\mathit{Enc}(\mathtt{ls}_i^j, \mathit{la\_id}_i \parallel c), \mathtt{ls}_i^j\}$ and the other of the form $\{\mathit{Enc}(\mathtt{ls}_i^j, \mathit{la\_id}_i \parallel c'), \mathtt{ls}_i^j\}$, for the same group of 2^n keys $\mathtt{ls}_i^j$; each table leads to a different set of candidates for $\mathtt{ls}_i(t)$, and candidates not appearing in both sets can be eliminated.

6.2 Birthday attacks on linkage seeds

A second attack that can be perpetrated against SCMS, aimed specifically at its forward privacy property, builds upon the fact that the k-bit long linkage seeds are computed via iterative hashing, using a fixed prefix for each LA, i.e., $\mathtt{ls}_i(t) = \mathit{Hash}(\mathit{la\_id}_i \parallel \mathtt{ls}_i(t-1))$. More precisely, to discover $\mathtt{ls}_i(t < t_s)$ from a given $\mathtt{ls}_i(t_s)$ placed in a CRL, an attacker can proceed as follows. First, the attacker picks 2^n random values $lc_{i,\alpha}(0)$, where n is a chosen parameter and $0 \leqslant \alpha < 2^n/\tau$. Each $lc_{i,\alpha}(0)$ is then used as the anchor for a hash chain of the form $lc_{i,\alpha}(j) = \mathit{Hash}(\mathit{la\_id}_i \parallel lc_{i,\alpha}(j-1))$, where $1 \leqslant j < \tau^*$ and τ^* is the length arbitrarily chosen by the attacker for its chains. For example, the attacker could set $\tau^* = w \times \tau$ for a small w, so the created chains' length would be close to the length of hash chains from the target LA_i. For simplicity, we assume that all $lc_{i,\alpha}(j)$ computed in this manner are distinct for any j and α, i.e., that no collisions occur; in this case, the attacker obtains 2^n hash chains at the cost of $\tau^* \times 2^n$ hash computations. In practice, though, collisions could be handled simply by merging the corresponding hash chains, so the total number of chains would be smaller and some of them would be longer than others.

Once again due to the birthday paradox, an attacker who gathers 2^m linkage seeds computed by LA_i has a high probability of finding a match between at least one of those linkage seeds and some previously computed $lc_{i,\alpha}(j)$ if $m + n + \lg(\tau^*) \geqslant k$. If a match occurs for $\mathtt{ls}_i(t_s)$ and $lc_{i,\alpha}(j)$, then a previous linkage seed $\mathtt{ls}_i(t_s - \epsilon)$ will also match $lc_{i,\alpha}(j-\epsilon)$. Assuming $lc_{i,\alpha}(j-\epsilon)$ is actually the pre-image of $\mathtt{ls}_i(t_s - \epsilon + 1)$, and not a second pre-image, this would allow the attacker to associate non-revoked certificates to the same device and, thus, violate the system's forward privacy.

This attack can be performed both by internal and external entities, using pre-computed hash chains for selected LAs; after all, it requires only access to linkage seeds from (public) CRLs and to the LAs' identifiers. Since the 2^m linkage seeds employed for this purpose can refer to any device and time period, once again the system's security degrades as time passes and certificates from devices served by the same LA_i are included in CRLs. Thus, for security reasons, the lifespan of a given LA_i may become limited by the choice of k and by the number of devices revoked with the participation of LA_i.

6.3 Protection against birthday attacks: security strings

One straightforward approach for addressing the birthday attacks described in Sections 6.1 and 6.2 would be to increase the linkage seed size. For example, one could adopt 192-bit linkage seeds, thus matching the second lowest key size supported by the AES block cipher. In that case, even after gathering 2^{64} linkage seeds or pre-linkage values, the attack's cost would still be $2^{192-64} = 128$. We note, however, that this approach does not actually eliminate the aforementioned security degradation. Instead, it only delays this issue, preventing the system from reaching a low security level too quickly. Furthermore, larger key sizes also lead to some undesirable overheads. Specifically, a total of 16 bytes added to each CRL entry (8 bytes per linkage seed); in addition, computing any $\mathtt{plv}$ would be approximately 20% more processing-consuming, since AES-128 and AES-192 encryption involve, respectively, 10 and 12 rounds. Hence, this simplistic solution is actually sub-optimal both in terms of security and performance.

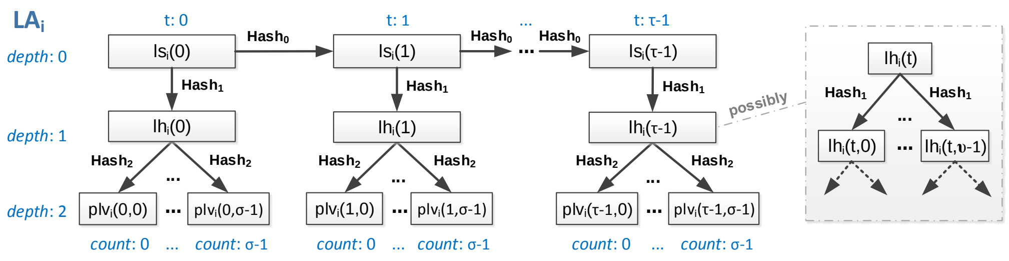

Figure 5: A more secure certificate linkage tree: adding security strings.

Table 2: Components of security strings

Field	Suggested length (bits)	Description
depth	8	Node's depth in tree (all linkage seeds are at depth 0, as shown in Fig. 5)
count	8	Node's index in time period and depth (starting at 0, as shown in Fig. 5)
t	24	Time period to which the node is associated
tree_id	40	Tree identifier (unique per tree from a given LA)
la_id	16	LA's identifier
Total	96	Whole security string

To address birthday attacks in a more effective and efficient manner, preventing any security degradation rather than masking it, we propose a slightly different process for building linkage trees. Our proposal consists basically in using hash functions whose inputs include a "security string", i.e., a different suffix for each hash function invocation, for the generation of pre-linkage values, linkage seeds, and linkage hooks (if the latter are present). As originally discussed by Leighton and Micali [34] in the context of hash-based signatures [35], such security strings limit the attackers' ability to use the birthday paradox in their favor, effectively thwarting the aforementioned birthday attacks. When applied to SCMS's linkage tree, the security string I can be built taking the tree's structure into account, as shown in Fig. 5. More precisely, each node receives a different value of I based on the linkage tree's identifier and on its location inside that tree. For this purpose, the fields shown in Table 2 can be employed, leading to $I = la\_id \,\|\, tree\_id \,\|\, t \,\|\, count \,\|\, depth$.

Using this security string, the linkage seeds, linkage hooks and pre-linkage values are computed by LA$_i$ as follows:

- Linkage seeds: $\mathtt{ls}_i(0)$ is picked at random, and $\mathtt{ls}_i(t) = Hash(\mathtt{ls}_i(t-1) \,\|\, la\_id \,\|\, tree\_id \,\|\, t-1 \,\|\, 0 \,\|\, 0)$.
- Linkage hooks: they are all computed as $\mathtt{lh}_i(t) = Hash(\mathtt{ls}_i(t) \,\|\, la\_id \,\|\, tree\_id \,\|\, t \,\|\, 0 \,\|\, 1)$.
- Pre-linkage values: they are all computed as $\mathtt{plv}_i(t, c) = Hash(\mathtt{lh}_i(t) \,\|\, la\_id \,\|\, tree\_id \,\|\, t \,\|\, c \,\|\, 2)$.

We emphasize that this approach is generic enough to accommodate further changes to the linkage tree's structure, simply by modifying the composition of the security strings. For example, suppose that more intermediate linkage hooks are added to give support to additional key usages, as discussed in Section 5. In this case, the security string's *count* parameter is simply adjusted to the linkage hook's position in the tree.

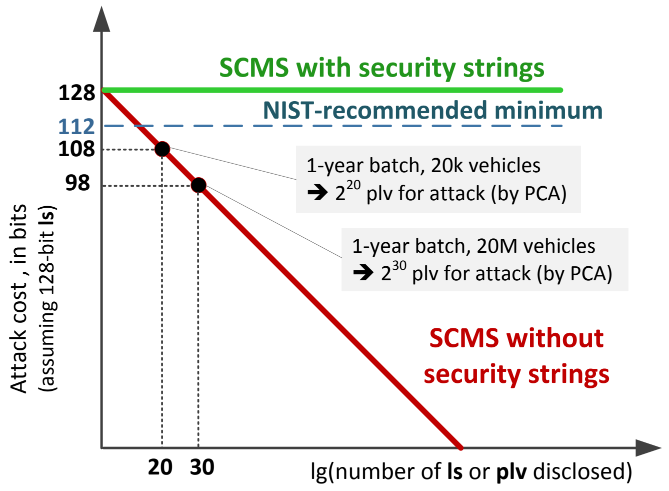

Figure 6: Long-term protection of security strings against birthday attacks.

6.4 Security analysis

Against the approach hereby proposed, the birthday attacks described in Sections 6.1 and 6.2 would proceed as follows. First, the attacker builds a large table containing entries of the form $\{h^j = Hash(str^j, I), str^j\}$ for a fixed security string I and for arbitrary k-long bit strings str^j, where $0 \leqslant j < 2^n$ for some n. Then, if some k-long linkage seed $\texttt{ls}_i(t)$ matches h^j, the attacker is able to recover the corresponding pre-image str^j, which should match $\texttt{ls}_i(t-1)$ with high probability. A similar reasoning applies if the match occurs for a linkage hook $\texttt{lh}_i(t)$ or for a pre-linkage value $\texttt{plv}_i(t,c)$, whose pre-images reveal $\texttt{ls}_i(t)$ and $\texttt{lh}_i(t)$, respectively.

Like before, the birthday paradox dictates that finding at least one match with high probability requires 2^m tests for $m + n \geqslant k$. Since I is used only once in the system, however, the attacker can only perform one test per I, meaning that the attack would work in practice only for $n \approx k$. If k is chosen appropriately (e.g., $k = 128$), the construction of such table with 2^k entries becomes computationally unfeasible. In addition, if the value of *tree_id* is unpredictable by attackers, they would not be able to pre-compute (parts of) such look-up table before one node in the corresponding tree is revoked. In this case, *tree_id* provides additional security similarly to what is done by salts in the context of password hashing [36]. Fig. 6 illustrates this long-term protection against birthday attacks, when compared to the original SCMS.

Finally, it is interesting to note that SCMS does use something similar to a security string in its design. Namely, it includes *la_id* in the derivation process employed for creating pre-linkage values and linkage seeds. This avoids security issues that might arise from collisions among data produced by different LAs. Therefore, the approach hereby proposed can be seen as an extension of the SCMS design, improving even further its resilience against birthday attacks.

6.5 Performance considerations

Adding a security string I to the computation of the linkage tree's nodes has little impact on processing as long as the hash function's input fits its block size. For example, SHA-256 operates on 512-bit blocks, appending at least 65 bits to its input message (a bit '1' for padding, and a 64-bit length indicator) [15]; therefore, a single call to its underlying compression function is enough to process a 128-bit linkage seed, linkage hook or pre-linkage value even when it is combined with a 319-bit security string.

On the other hand, parts of the security string need to be published in the CRLs together with the corresponding linkage seeds or linkage hooks. Therefore, its length should be limited to avoid unnecessary communication overheads, motivating the parameter lengths suggested in Table 2. Specifically, the *depth* and *count* fields correspond to counters, so they do not need to be explicitly included in the CRL: it suffices to indicate whether the entry refers to a linkage seed or a linkage hook to determine whether $depth = 0$ or $depth = 1$, respectively, and in both cases *count* starts at zero. The LA's identifier *la_id* and the time period t are likely to be explicitly shown, although multiple CRL entries in which these fields are identical may be grouped together under the same data structure, thus avoiding the need of repeating them. In addition, those fields would also have to be shown in SCMS, so they do not represent any extra overhead when compared to the original scheme. In [9], the suggested length of *la_id* is 16 bits, hence the size hereby proposed.

The *tree_id* field is, thus, the only actual added by the proposed solution when compared to SCMS. The reason is that it should be unique for each CRL entry from the same LA, so it always appears explicitly. The choice of 40 bits for this field is justified by the fact that this is the length suggested in the literature for vehicle identifiers [8]. Therefore, this length should be enough for indexing any linkage tree ever created. Meanwhile, it prevents the system's security degradation with at most 62.5% of the overhead resulting from the sub-optimal solution of increasing the system's key sizes. Actually, in some cases this overhead does not even need to be exactly 40 bits per linkage seed in CRL entry. For example, if *tree_id* assumes the form of a serial number starting at 0, one can omit or compress its leading zeros to save space in CRLs. Alternatively, *tree_id* might be built by the RA itself as a concatenation of (1) its own identifier, (2) a serial number and (3) one bit $b = 0$ or $b = 1$ for discerning between LA_1 and LA_2, respectively. In that case: the RA can ensure that each LA's *tree_id* remains unique, simply by using different serial numbers for different vehicles; the serial number can replace the two values of *tree_id* (one for each LA) that, otherwise, would need to be placed in the CRL; and the bit b can be inferred simply by the order in which the LA's information (i.e., linkage hooks or seeds) are placed in the CRL. The result would then be 40 bits per CRL entry, which is 31.25% of the 16 bytes incurred in a solution relying on larger key sizes.

As a final remark, it is interesting to note that one can obtain a zero-overhead solution by simply omitting *tree_id* entirely. Even though this extreme approach would be unable to fully prevent birthday attacks, the resulting system's security would still be higher than what is provided by SCMS. The reason is that, since t remains as part of the hash function's input, any table mapping hashes to their pre-images would be bound to a specific time period. More precisely, suppose that some hash $h(t_1)$, computed from t_1 and part of the attacker's table, matches a linkage seed $\texttt{ls}_i(t_2)$ from a vehicle. If $t_1 \neq t_2$, then the pre-images of $h(t_1)$ and $\texttt{ls}_i(t_2)$ inevitably differ, since the former is computed from $t_1 - 1$ and the latter from $t_2 - 1$. An analogous argument applies to a linkage hook $\texttt{lh}_i(t_2)$ and to a pre-linkage value $\texttt{plv}_i(t_2, c)$, the difference being that the pre-images for both would involve t_2, whereas $h(t_1)$ would take t_1. As a result, the system's security only degrades with the number of *vehicles* whose linkage trees share the same time period, not with the size of linkage trees as in the original SCMS. This also means that, unlike in SMCS, attacking any given time period would require the construction of a distinct pre-image table, forcing attackers to divide their resources if their goal is to attack multiple time periods.

7 Linking certificates without Linkage Authorities

In the original SCMS design, even though a single LA cannot identify which set of certificates belong to the same device, it is trivial for LAs in collusion to do so: after all, they are responsible for creating and storing pre-linkage values, so these entities can easily compute a certificate's linkage value by combining their knowledge. This is actually the basis of SCMS's efficient revocation process, so at first sight this might be seen as unavoidable given the scheme's design goals. Unfortunately, however, this represents an extra point of collusion in the VPKI, in addition to the one already presented by PCAs colluding with RAs during the creation of certificates (independently of any linkage values thereby enclosed). Even worse: as described in this section, there are many possible ways by which a dishonest RA or PCA can collude with a single LA and, as a result, gain the ability to track vehicles. The solution hereby presented, which eliminates LAs as independent entities, is aimed at reducing this attack surface while preserving the system's ability to revoke/linking misbehaving vehicles.

7.1 Privacy issues involving LAs

Besides the straightforward collusion between LAs for learning linkage values, there are a few other ways to exploit their existence as separate entities aiming to violate the system's privacy. Namely, we describe three possible attacks: replay attacks by RA, RA-LA collusion and PCA-LA collusion.

- *Replay attacks by RA*: as described in Section 4.2, in SCMS the RA is responsible for relaying the LA-generated pre-linkage values, $\texttt{plv}_1$ and $\texttt{plv}_2$, to the PCA. Even though both pre-linkage values are encrypted, so only the PCA learns their actual values, the RA can still replay a $(\texttt{plv}_1, \texttt{plv}_2)$ pair for some (or all) pseudonym certificates from a target vehicle. If the PCA does not employ any mechanism for identifying the freshness of such (encrypted) pre-linkage values, the corresponding certificates would receive the same $\texttt{lv}$. Hence, a simple relationship is created between those certificates. Fortunately, such a relationship is obvious enough to be detectable by vehicles, which might refuse to use certificates that have the same $\texttt{lv}$: after all, for k-bit long linkage values, such collisions occur only with a negligible probability when the number of certificates is much smaller than $2^{k/2}$. Nevertheless, variants of this attack would be harder to detect. In particular, suppose the RA sends bogus pseudonym certificate requests to the PCA, containing (1) cocoon keys for which the RA knows the decryption key, and (2) the same encrypted $(\texttt{plv}_1, \texttt{plv}_2)$ pair that will be used later when requesting certificates for a target vehicle. By decrypting the bogus certificate, the RA learns the linkage value $\texttt{lv}$ that is placed on the target's certificate, and can then associate that certificate to its owner. The RA might even collect raw pre-linkage values, simply by replacing $\texttt{plv}_1$ with a $\texttt{plv}_*$ that was previously revealed (e.g., due to a revocation); by doing so, the companion $\texttt{plv}_2$ in the bogus certificate request can be computed as $\texttt{plv}_2 = \texttt{lv} \oplus \texttt{plv}_*$. Unless the PCA filters such replayed pre-linkage values, the vehicle itself would be unable to defend from (or even detect) such attacks. Standard methods for checking data freshness, however, may encumber the batch generation process: if timestamps are used, an honest RA might be prevented from collecting many $\texttt{plv}$ values from the LAs for accelerating the certificate batching generation; if data freshness is enforced via nonces, on the other hand, the PCA would need to store a large number of nonces for each LA.

- *RA-LA collusion*: by colluding with LA_1, the RA can provide a fresh $\texttt{plv}_1$ and replay a known value $\texttt{plv}_*$ from LA_2. In that case, every linkage value $\texttt{lv}(t, c)$ in the target certificate takes the form $\texttt{lv}(t, c) = \texttt{plv}_1(t, c) \oplus \texttt{plv}_*$. Hence, even though each $\texttt{lv}(t, c)$ appears to be uncorrelated from the vehicle's perspective, it would be easy for RA and LA_1 to identify certificates belonging to the same vehicle: it would suffice to check whether $\texttt{lv}(t, c) \oplus \texttt{plv}_*$ results in values of $\texttt{plv}_1(t, c)$ that belong to

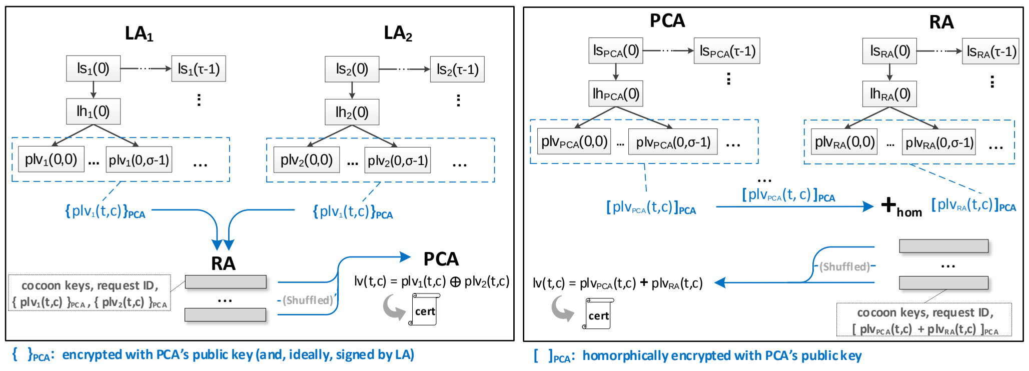
LA₁
ls₁(0)
ls₁(τ-1)
lh₁(0)
plv₁(0,0)
plv₁(0,σ-1)
LA₂
ls₂(0)
ls₂(τ-1)
lh₂(0)
plv₂(0,0)
plv₂(0,σ-1)
{plv₁(t,c)}PCA
{plv₂(t,c)}PCA
RA
cocoon keys, request ID, { plv₁(t,c) }PCA , { plv₂(t,c) }PCA
(Shuffled)
PCA
lv(t,c) = plv₁(t,c) ⊕ plv₂(t,c)
cert
{ }PCA: encrypted with PCA's public key (and, ideally, signed by LA)
PCA
lsPCA(0)
lsPCA(τ-1)
lhPCA(0)
plvPCA(0,0)
plvPCA(0,σ-1)
RA
lsRA(0)
lsRA(τ-1)
lhRA(0)
plvRA(0,0)
plvRA(0,σ-1)
[plvPCA(t,c)]PCA
[plvPCA(t, c)]PCA
+hom
[plvRA(t,c)]PCA
(Shuffled)
lv(t,c) = plvPCA(t,c) + plvRA(t,c)
cert
cocoon keys, request ID, [plvPCA(t,c) + plvRA(t,c)]PCA
[ ]PCA: homorphically encrypted with PCA's public key

Figure 7: Using linkage values in SCMS (left) and in the LA-free approach proposed in this article (right).

the same linkage tree. Once again, preventing such attack would require the PCA to check the freshness of the $(\mathtt{plv}_1, \mathtt{plv}_2)$ pair.

- *PCA-LA collusion*: the PCA could collude with LA$_1$ to identify which values of $\mathtt{plv}_1$ belong to the same linkage tree upon the generation of the corresponding pseudonym certificates. Like an RA-PCA collusion, this attack would remain undetectable by any system entity or vehicle.

7.2 Generating linkage values without LAs

The proposed LA-free approach still relies on linkage values for correlating certificates belonging to the same user, similarly to what is done in the original SCMS (see Fig. 7). The linkage trees employed may encompass only linkage seeds and pre-linkage values, as in Fig. 3, or also include linkage hooks for allowing temporary revocation of vehicles, as depicted in Fig. 5. Whichever the case, however, it is the PCA who is responsible for generating its own linkage trees, playing a role similar to LA$_1$ in the system. Each pre-linkage value in this tree is then encrypted by the PCA with its public key, using an additively homomorphic encryption algorithm (e.g., the Paillier cryptosystem [37]). Those pre-linkage values are associated to the same identifier $tree\_id_i^{PCA}$, unique per tree, and are then sent to the RA. As a result, the RA is able to recognize which set of pre-linkage values $\{\mathtt{plv}_{PCA}(t,c)\}$ belong to the same linkage tree, and also identify their corresponding indices in that tree, t and c. Nevertheless, the RA is unable to decrypt any given $\mathtt{plv}_{PCA}(t,c)$, so it never learns any actual pre-linkage value.

Similarly, the RA also creates a set of linkage trees, playing a role analogous to LA$_2$. Then, whenever a new batch of certificates needs to be issued, the RA selects one of its own trees, identified by $tree\_id_i^{RA}$, and homomorphically adds its corresponding pre-linkage values to those received from the PCA. The outcome is that the RA obtains a set of (encrypted) linkage values of the form $\mathtt{lv}(t,c) = \mathtt{plv}_{PCA}(t,c) + \mathtt{plv}_{RA}(t,c)$, which can only be decrypted by the PCA. The RA then delivers to the PCA those (encrypted) linkage values, together with any other data normally included in the pseudonym certificate issuance process (e.g., cocoon keys), using SCMS's shuffling mechanism for mixing up requests from different users. Finally, the PCA creates and signs the certificates as usual, the only difference being that $\mathtt{lv}(t,c)$ is retrieved directly from the RA's request, instead of computed by XORing pre-linkage values provided by different LAs.

As a result of this process, even though PCA and RA create the pre-linkage values without the intervention of any LA, they have no knowledge of which pre-linkage value is attached to each certificate (unless, of course, they collude). More precisely, the RA does not learn any $\texttt{plv}_{PCA}(t,c)$ received from the PCA, since they are random-like values encrypted with PCA's public key; hence, the RA is unable to determine $\texttt{lv}(t,c)$ despite the fact that its computation involves the (known) pre-linkage value $\texttt{plv}_{RA}(t,c)$. The PCA, in turn, is unable to determine which $\texttt{plv}_{PCA}(t,c)$ corresponds to a given $\texttt{lv}(t,c)$ received from the RA, since $\texttt{plv}_{RA}(t,c)$ acts as a random mask during the computation of $\texttt{lv}(t,c)$; therefore, assuming that the RA correctly shuffled the vehicles' requests, any received $\texttt{lv}(t,c)$ follows a uniform distribution from the PCA's perspective. Consequently, as in the original SCMS, the RA only knows that a given batch of certificates belongs to the same user, but has no access to the batch's contents, including the linkage values enclosed in the certificates. In comparison, the PCA knows the certificates' contents, but cannot link any information thereby enclosed to a given user; for example, it is unable to correlate a $\texttt{lv}(t,c)$ to its corresponding $\texttt{plv}_{PCA}(t,c)$ and, thus, to a specific linkage tree.

7.3 The LA-free revocation process

Revocation in the proposed LA-free scheme follows a process quite similar to the one originally proposed in SCMS [7]. Namely, when the MA detects that the owner of a given certificate *cert* is misbehaving, it provides that certificate's linkage value, $\texttt{lv}$, to the PCA. In response, the PCA sends to the MA the identifier of the request in which *cert* was generated. The MA can then ask the RA for the values of $\texttt{plv}_{RA}$ and $\texttt{plv}_{PCA}$ employed in that request: the former is known by the RA, so its actual value can be presented together with any additional data that allows associated certificates to be revoked in a forward secure manner (e.g., a linkage seed or hook, depending on whether the revocation should be permanent or temporary); the latter, on the other hand, is homomorphically encrypted with the PCA's public key, so only this encrypted data is provided by the RA. With this information, the MA obtains from the PCA the decrypted value of $\texttt{plv}_{PCA}$, together with the corresponding linkage seed/hook that must be disclosed as part of the revocation process. Finally, the MA places those linkage seeds/hooks in a CRL, so anyone can use them in the computation of $\texttt{lv}(t_s,c)$, for any revoked time period t_s. The RA can also place the corresponding vehicle in a blacklist, so it cannot receive new certificates.

When compared with the original SCMS revocation procedure, the main difference in the described process is that the PCA needs to be contacted twice, the first for identifying the pseudonym certificate request and the second for the retrieval of the unencrypted value of $\texttt{plv}_{PCA}$. In addition, this process is designed to avoid the leakage of information between PCA and RA, as well as to allow extra verifications by the MA. More precisely, the MA should confirm that the correct certificate is being revoked, by checking that $\texttt{lv} \overset{?}{=} \texttt{plv}_{RA} + \texttt{plv}_{PCA}$ holds true and that the provided linkage seed/hook does produce those linkage values. Hence, if either PCA or RA sends an invalid seed/hook to the MA either due to an unintentional mistake or to malicious intent (e.g., an attempt to prevent the vehicle from being revoked), this issue can be detected. The reason is that, since the PCA (resp. RA) does not learn the value of $\texttt{plv}_{RA}$ (resp. $\texttt{plv}_{PCA}$) in this process, providing a wrong value of $\texttt{plv}_{PCA}$ (resp. $\texttt{plv}_{RA}$) should lead to the correct $\texttt{lv}$ with negligible probability.

7.4 Detection of dishonest RA by MA

In principle, one possible drawback of the proposed solution is that it does not prevent a dishonest RA from providing a bogus linkage value to the PCA aiming to track devices. More precisely, suppose that the RA does not use the additively homomorphic scheme to compute the encrypted value of $\texttt{lv}(t,c) = \texttt{plv}_{RA}(t,c) + \texttt{plv}_{PCA}(t,c)$. Instead, it simply encrypts an arbitrary bitstring $z(t,c)$ with the PCA's public key, and then presents the resulting ciphertext in place of the correct $\texttt{lv}(t,c)$. By design, the PCA should be unable to distinguish a correctly computed $\texttt{lv}(t,c)$ from a random bitstring, since otherwise it might also be able to identify which value of $\texttt{plv}_{PCA}(t,c)$ was employed in the computation of $\texttt{lv}(t,c)$ (and, thus, to associate different requests to the same user). Therefore, such trickery would go unnoticed by the PCA, and $z(t,c)$ would be used as that certificate's linkage value. Then, it would be trivial for the RA, who knows every value of $z(t,c)$ and also the identities of the vehicles making each request, to link a pseudonym certificate to its owner.

Besides violating the users' privacy, such misbehavior from the RA might have disastrous consequences to the VPKI's revocation process. Namely, it would prevent an MA from actually revoking the corresponding certificates using a single pair of linkage seeds/hooks. The reason is that it is very unlikely that the set of $(\texttt{plv}_{RA}(t,c), \texttt{plv}_{PCA}(t,c))$ derived from the RA's and PCA's linkage seeds/hooks would match the arbitrary $z(t,c)$ values inserted into the certificates as linkage values. Actually, except for a negligible probability, this should only happen if every $z(t,c)$ value was originally computed from a linkage seed/hook, like $\texttt{plv}_{RA}(t,c)$, and then added to the corresponding $\texttt{plv}_{PCA}(t,c)$.

This issue is taken into account in the revocation procedure described in Section 7.3, which enables the MA to identify that something is wrong when the linkage seeds/hooks provided by PCA and RA do not lead to the expected $\texttt{lv}(t,c)$. Indeed, when the RA forces $\texttt{lv}(t,c) = z(t,c)$ aiming to track vehicles, it would only pass the $\texttt{lv}(t,c) \stackrel{?}{=} \texttt{plv}_{PCA}(t,c) + \texttt{plv}_{RA}(t,c)$ check performed by the MA if: (1) the RA is able to provide $z(t,c) - \texttt{plv}_{PCA}(t,c)$ as the value of $\texttt{plv}_{RA}(t,c)$, as well as linkage seeds/hooks that are pre-images of such $\texttt{plv}_{RA}(t,c)$; or (2) $\texttt{plv}_{PCA}(t,c) = 0$ for every t and c, in which case the RA can simply compute $z(t,c)$ from a regular linkage tree and provide linkage seeds/hooks as usual. However, since RA never learns the value of $\texttt{plv}_{PCA}$ during the pseudonym certificate issuing process, it cannot compute $z(t,c) - \texttt{plv}_{PCA}(t,c)$, let alone find the corresponding pre-images; in addition, the $\texttt{plv}_{PCA}(t,c) = 0$ condition only occurs with negligible probability, since each $\texttt{plv}_{PCA}(t,c)$ is the output of a hash function. Hence, it is unfeasible for a malicious RA to avert such misbehavior detection by the MA whenever a certificate is revoked.

The security of such an auditing procedure can be more formally summarized in Theorem 1.

Theorem 1. Detection (by MA) of invalid linkage value provided by RA*: Let $\mathcal{E}$ be a homomorphic encryption algorithm, and let $\mathcal{E}(\texttt{plv}_{PCA})$ be the result of encrypting a secret $\texttt{plv}_{PCA}$ with $\mathcal{E}$. Suppose that, given $\mathcal{E}(\texttt{plv}_{PCA})$, a malicious RA produces $\mathcal{E}(\texttt{lv}) = \mathcal{E}(z)$, for an arbitrary z. Then, that RA is able to provide an arbitrary $\texttt{plv}_{RA}$ that passes the check $\texttt{lv} \stackrel{?}{=} \texttt{plv}_{PCA} + \texttt{plv}_{RA}$ if and only if that RA is also able to violate the confidentiality of $\mathcal{E}$.*

Proof. Since in this scenario the RA ends up forcing $\texttt{lv} = z$, passing the check $\texttt{lv} \stackrel{?}{=} \texttt{plv}_{PCA} + \texttt{plv}_{RA}$ implies $z = \texttt{plv}_{PCA} + \texttt{plv}_{RA}$. Hence, if the RA is able to choose z and $\texttt{plv}_{RA}$ satisfying this equation, this also means that the RA can compute $\texttt{plv}_{PCA} = z - \texttt{plv}_{RA}$ given $\mathcal{E}(\texttt{plv}_{PCA})$. Similarly, if the RA is able to obtain the secret $\texttt{plv}_{PCA}$ from $\mathcal{E}(\texttt{plv}_{PCA})$, it is straightforward to pick z and $\texttt{plv}_{RA}$ accordingly. □

7.5 Detection of dishonest RA by PCA

Albeit useful, the mechanism described in Section 7.4 for misbehavior detection by an MA is only applicable when a CRL is issued. Therefore, it may not be enough in practice, for at least two reasons: first, honest users are not expected to be revoked, meaning that they could be inconspicuously tracked by the malicious RA for the entire lifetime of their certificates; second, some recent proposals preclude the need of certificate revocation, but instead focus on preventing their decryption by misbehaving vehicles [8][11], in which case the RA's honesty would be rarely (or never) scrutinized.

Fortunately, an auxiliary mechanism can be employed by the PCA for a more frequent evaluation of an RA's behavior. Namely, without loss of generality, assume that an RA should be audited periodically, after a total of n pseudonym certificates $cert_i$ (where $1 \leqslant i \leqslant n$) are issued by the PCA. Whenever this number is reached, the PCA requests (1) the sum of the n pre-linkage values generated by the RA for those certificates, denoted $\theta_{RA} = \sum_{i=1}^{n}(\texttt{plv}_{RA,i})$; and (2) the shuffled list of all encrypted $\texttt{plv}_{PCA,i}$ associated to those certificates — or, equivalently, a shuffled list containing the IDs of the corresponding PCA's linkage trees, as well as the indices of every $\texttt{plv}_{PCA,i}$ in those trees. With this information, the PCA sums up its own pre-linkage values $\texttt{plv}_{PCA,i}$, obtaining $\theta_{PCA} = \sum_{i=1}^{n}(\texttt{plv}_{PCA,i})$, without learning in which order each $\texttt{plv}_{PCA,i}$ was used by the RA. The PCA also adds together the corresponding linkage values that were inserted in the n certificates issued during that period, obtaining $\sum_{i=1}^{n}(\texttt{lv}_i)$. Finally, the PCA checks whether $\theta_{RA} + \theta_{PCA} \overset{?}{=} \sum_{i=1}^{n}(\texttt{lv}_i)$; the certificates were created properly only if this equality holds true.

Similarly to the MA's verification procedure, this auxiliary auditing mechanism performed by the PCA allows the latter to identify situations in which the RA responds with an arbitrary z_i instead of using $\texttt{plv}_{PCA,i}$ for computing the correct (homomorphically encrypted) linkage value $\texttt{lv}_i$. More precisely, by misbehaving in this manner, the RA ends up forcing $\texttt{lv}_i = z_i$ and, hence, $\sum_{i=1}^{n}(z_i) = \sum_{i=1}^{n}(\texttt{lv}_i)$. Therefore, the RA would only be able to avert detection if it provides in its response a value of $\theta_{RA*} = \sum_{i=1}^{n}(\texttt{plv}_{RA,i}) + \sum_{i=1}^{n}(\texttt{plv}_{PCA,i})$, which should be unfeasible because the RA never learns the value of any $\texttt{plv}_{PCA,i}$ during the pseudonym certificate issuing process. The main difference when compared with the MA's procedure is that this process prevents the PCA from learning which $\texttt{plv}_{PCA,i}$ is associated with each certificate, so it cannot track devices either. More precisely, the PCA only learns which pre-linkage values have already been used and, thus, can estimate how many vehicles have received the different $cert_i$. As long as the number of vehicles is large enough, however, this knowledge does not lead to any actual privacy issue.

Theorem 2. Detection (by PCA) of invalid linkage value provided by RA*: Let $\mathcal{E}$ be a homomorphic encryption algorithm, and let $\mathcal{E}(\texttt{plv}_{PCA,i})$ be the result of encrypting a secret $\texttt{plv}_{PCA,i}$ with $\mathcal{E}$. Suppose that, given a set containing n encrypted pre-linkage values $\mathcal{E}(\texttt{plv}_{PCA}, i)$(say, for $1 \leqslant i \leqslant n$), a malicious RA selects some of them and responds: (1) honestly, with $\mathcal{E}(\texttt{lv}_i) = \mathcal{E}(\texttt{plv}_{PCA,i} + \texttt{plv}_{RA,i})$; and dishonestly, with $\mathcal{E}(\texttt{lv}_i) = \mathcal{E}(z_i)$, for arbitrary values of z_i. Then, that RA is able to provide an arbitrary θ_{RA} that passes the check $\theta_{RA} + \sum_{i=1}^{n}(\texttt{plv}_{PCA,i}) \overset{?}{=} \sum_{i=1}^{n}(\texttt{lv}_i)$ if and only if that RA is also able to violate the confidentiality of $\mathcal{E}$.*

Proof. This can be seen as a corollary of Theorem 1. Specifically, let H and D denote the sets of indices for which the RA responds honestly and dishonestly, respectively. The verification equation can then be divided considering those sets, so:

$$\theta_{RA} + \textstyle\sum_{i=1}^{n}(\texttt{plv}_{PCA,i}) \overset{?}{=} \sum_{i=1}^{n}(\texttt{lv}_i)$$

$$\theta_{RA} + \textstyle\sum_{h\in H}(\texttt{plv}_{PCA,h}) + \sum_{d\in D}(\texttt{plv}_{PCA,d}) \overset{?}{=} \sum_{h\in H}(\texttt{lv}_h) + \sum_{d\in D}(\texttt{lv}_d)$$

Replacing the RA's responses for each set H and D, we have:

Table 3: Performance of LA-free solution compared to original SCMS.

	Operation	Time (cycles)	Time (sec.)	Phase
LA-free	Paillier keygen (PCA)	$1.4\times10^{+9}$	0.4	Bootstrap
	`plv` encryption (PCA,RA)	$114.0\times10^{+6}$	31.7×10^{-3}	Out-of-band
	Homomorphic add (RA)	$0.03\times10^{+6}$	0.01×10^{-3}	In-band
	`lv` decryption (PCA)	$17.7\times10^{+6}$	4.9×10^{-3}	
SCMS	ECIES keygen (PCA)	$0.2\times10^{+6}$	0.06×10^{-3}	Bootstrap
	`plv` encryption (LAs)	$0.2\times10^{+6}$	0.05×10^{-3}	Out-of-band
	2 `plv` decryption (PCA)	$0.4\times10^{+6}$	0.1×10^{-3}	In-band

$$\theta_{RA} + \textstyle\sum_{h\in H}(\texttt{plv}_{PCA,h}) + \sum_{d\in D}(\texttt{plv}_{PCA,d}) \overset{?}{=} \\ \textstyle\sum_{h\in H}(\texttt{plv}_{PCA,h}) + \sum_{h\in H}(\texttt{plv}_{RA,h}) + \sum_{d\in D}(z_d)$$

$$\theta_{RA} + \textstyle\sum_{d\in D}(\texttt{plv}_{PCA,d}) \overset{?}{=} \sum_{h\in H}(\texttt{plv}_{RA,h}) + \sum_{d\in D}(z_d)$$

In principle, every $\texttt{plv}_{RA,h}$ and z_d are under the attacker's control, and so is θ_{RA}. However, if the RA can somehow find a value of θ_{RA} that satisfies this equation, this implies that RA's ability to compute $\sum_{d\in D}(\texttt{plv}_{PCA,d}) = \sum_{h\in H}(\texttt{plv}_{RA,h}) + \sum_{d\in D}(z_d) - \theta_{RA}$ given every value of $\mathcal{E}(\texttt{plv}_{PCA,d})$. In other words, this means that the RA is able to violate the confidentiality of $\mathcal{E}$ for the ciphertext $\mathcal{E}(\sum_{d\in D}(\texttt{plv}_{PCA,d}))$, which can be obtained from the individual $\mathcal{E}(\texttt{plv}_{PCA,d})$ using the additively homomorphic property of $\mathcal{E}$.

The converse implication follows an analogous reasoning. After all, as long as the malicious RA can decrypt $\mathcal{E}(\sum_{d\in D}(\texttt{plv}_{PCA,d}))$, it is trivial to compute a valid θ_{RA} by combining the obtained plaintext with the previously picked $\texttt{plv}_{RA,h}$ and z_d. □

Finally, it is noted that an analogous perusal by the RA to verify the PCA's honesty is unnecessary, since the PCA has no advantage in misbehaving during the computation of linkage values. For example, if the PCA inserts an arbitrary value z in the certificate instead of decrypting and using the `lv` provided by the RA, this would not reveal any information about the owner of the certificates. Instead, it would only needlessly disrupt the revocation process in a manner that is detectable by the MA when, as discussed in Section 7.4, the latter verifies whether $\texttt{lv} \overset{?}{=} \texttt{plv}_{RA} + \texttt{plv}_{PCA}$ holds true. Actually, a similar non-issue occurs in the original SCMS protocol, in which the PCA might replace the linkage values received from the LAs by an arbitrary bitstring. Similarly to the LA-free solution, (1) the PCA would not gain any knowledge by doing so and (2) the PCA could be detected if the MA performs a similar verification considering the pre-linkage values provided by the LAs.

7.6 Performance and cost analysis

To assess the performance of the proposed LA-free solution, we benchmarked its main building blocks and compared the results with those obtained with similar-purpose operations in the original SCMS. Specifically, for homomorphic encryption and decryption, we employed the Paillier cryptosystem with 3072-bit keys, which provides 128-bit security; for this, we used the paillier v0.2.0 Rust library [38]. For the regular encryption and decryption in SCMS, we use the ECIES [14] cryptosystem built upon NaCl library [39], with Curve-25519 [40], once again getting a 128-bit security level.

For the proposed LA-free solution, we measured the time taken by the following operations: key generation by PCA, which is performed only once as part of the system's initialization; encryption of pre-linkage values by RA and PCA, which can be performed out-of-band, before pseudonym certificates are actually requested; and the homomorphic operation of pre-linkage values (by the RA) and the subsequent decryption (by the PCA), both of which are performed in-band, as part of the certificate issuance procedure. In SCMS, we consider: the (one-time) ECIES key generation by the PCA; the encryption of one pre-linkage value by each LA, which is assumed to be done out-of-band; and the decryption of two pre-linkage values by the PCA, which has to be performed in-band.

Table 3 shows the results obtained, both in cycles and in seconds, considering a single pseudonym certificate request on a machine equipped with an Intel Core i7-7700, at 3.60 GHz. The numbers correspond to the average cost of 250,000 executions for each operation, leading to a standard deviation below 7% in all cases. As shown in this table, most of the processing overhead in the proposed solution can be done out-of-band and, thus, does not actually impact the latency of certificate provisioning. The in-band processing of the LA-free solution, in turn, is dominated by the decryption procedure performed by the PCA, which is roughly 45× larger than the corresponding process in the original SCMS. Albeit relatively high, the incurred latency remains quite small: the total time is smaller 5 ms in our testbed. Running SCMS on more powerful servers or specialized hardware should lead to even lower numbers in practice. Also, in a large-scale deployment, this extra latency can be (at least partially) masked by processing different requests with parallel hardware.

In terms of bandwidth usage, the proposal also adds some overhead. Specifically, for a 128-bit security level, the data structures resulting from a Paillier encryption are roughly 6144-bit long. In comparison, ECIES should lead to approximately 448-bit encrypted packages for the same security level, assuming 64-bit authentication tags and 128-bit pre-linkage values. On the other hand, the absence of LAs reduces the number of communications in half (namely, there are no LA-to-RA transmissions). Therefore, the net cost of the proposal when compared with the original SCMS is 10 KiB per pseudonym certificate, or a 6.9× overhead. We emphasize, though, that this extra communication applies only to RA and PCA, which should have high-bandwidth capabilities, while resource-limited devices such as vehicles are not affected.

Finally, it is worth examining the actual financial costs for implementing the proposed LA-free solution, when compared to the original SCMS. At first sight, it might seem that our proposal translates to a more expensive VPKI, since extra hardware may be required for handling the higher processing at the PCA, as well as the extra bandwidth costs. However, in reality an LA-free solution is likely to create a *less* costly VPKI. After all, the secure facilities that would be required for implementing the PKI-like infrastructure for running LAs [27, Table 9], as well as the personnel for managing them, become unnecessary. This is especially relevant because the capital costs of PKIs are dominated exactly by the construction of secure premises (e.g., physical installations, access control and monitoring equipment, telecommunication and disaster recovery systems) [41]. Furthermore, their main yearly costs, which may reach hundreds of thousands or even millions of dollars for a scale much smaller than VPKIs, come from the maintenance of such facilities, regular audits, and personnel expenditures [41, 42]. Hence, in practice the extra hardware investments for handling additively homomorphic operations should be compensated by the financial savings on the VPKI itself, besides the reduced architectural complexity [43].

8 Conclusion

The broad adoption of intelligent transportation systems (ITS) requires attention to two important aspects: data authentication, so invalid messages can be filtered out by vehicles; and (revocable) user privacy, so honest drivers cannot be tracked by their peers or by the system itself. A promising design tackling such requirements is the Security Credential

Management System (SCMS), which provides efficient, scalable and privacy-preserving mechanisms for issuing and revoking pseudonym certificates. Such characteristics recently motivated standardization efforts [9], as well as research initiatives aiming to enhance SCMS's design [8][10][11].

In this paper, we provide three contributions for improving SCMS's linkage and revocation process. First, we propose modifications to the structure of SCMS's key linkage tree aiming to address additional use cases. In particular, the proposed approach efficiently supports the temporary revocation and linkage of pseudonym certificates, which should be useful to implement suspension mechanisms or aid in investigations by law-enforcement authorities. Second, we describe two birthday attacks against SCMS's key linkage/revocation process, both of which are able to affect the vehicles' privacy as time passes and more certificates are revoked. One way to tackle this issue would be to limit the lifespan of the corresponding Linkage Authorities (LAs), thus preventing the security level from dropping below a given threshold. Instead, the proposed approach based on security strings averts this security degradation with a small overhead, or, alternatively, increases the systems security without incurring any overhead. Finally, we show how the role of LAs can be redistributed to the PCA and RA in a secure manner, eliminating the need of maintaining LAs as independent entities. The result is improved security and lower deployment costs, despite a small increase in the total latency time for issuing pseudonym certificates.

Acknowledgment. This work was supported by the Brazilian National Council for Scientific and Technological Development (CNPq – grant 301198/2017-9), and by LG Electronics.

References

[1] 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," NHTSA, Tech. Rep. DOT HS 812 014, 2014.

[2] P. Papadimitratos, A. L. Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," *IEEE Communications Magazine*, vol. 47, no. 11, pp. 84–95, November 2009.

[3] NHTSA, "Federal Motor Vehicle Safety Standards; V2V Communication," National Highway Traffic Safety Administration, U.S. Department of Transportation (USDOT), Tech. Rep. 8, Jan 2017.

[4] M. Khodaei and P. Papadimitratos, "The key to intelligent transportation: Identity and credential management in vehicular communication systems," *IEEE Veh. Technol. Mag.*, vol. 10, no. 4, pp. 63–69, Dec 2015.

[5] P. Cincilla, O. Hicham, and B. Charles, "Vehicular PKI scalability-consistency trade-offs in large scale distributed scenarios," in *IEEE Vehicular Networking Conference (VNC)*, Dec 2016, pp. 1–8.

[6] J. Petit, F. Schaub, M. Feiri, and F. Kargl, "Pseudonym schemes in vehicular networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 228–255, 2015.

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

[8] V. Kumar, J. Petit, and W. Whyte, "Binary hash tree based certificate access management for connected vehicles," in *Conference on Security and Privacy in Wireless and Mobile Networks (WiSec'17)*. New York, NY, USA: ACM, 2017, pp. 145–155.

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

[10] M. Simplicio, E. Cominetti, H. Kupwade-Patil, J. Ricardini, and M. Silva, "The unified butterfly effect: Efficient security credential management system for vehicular communications," in *IEEE Vehicular Networking Conference (VNC'18)*, 2018.

[11] ——, "ACPC: Efficient revocation of pseudonym certificates using activation codes," *Ad Hoc Networks*, p. (in press), 2018.

[12] 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, pp. 1322–1329.

[13] NIST, *FIPS 197: Advanced Encryption Standard (AES)*, National Institute of Standards and Technology, November 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

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

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

[16] NIST, *FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, National Institute of Standards and Technology, August 2015, doi:10.6028/NIST.FIPS.202.

[17] ——, *FIPS 186-4: Digital Signature Standard (DSS)*, National Institute of Standards and Technology, July 2013. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[18] D. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, Sep 2012.

[19] S. Josefsson and I. Liusvaara, "RFC 8032: Edwards-curve digital signature algorithm (EdDSA)," https://tools.ietf.org/html/rfc8032, Jan 2017.

[20] ETSI, "TR 102 941 – intelligent transport systems (ITS); security; trust and privacy management," European Telecommunications Standards Institute, Tech. Rep., Jun 2012.

[21] IEEE, "IEEE standard for wireless access in vehicular environments–security services for applications and management messages - amendment 1," *IEEE Std 1609.2a-2017 (Amendment to IEEE Std 1609.2-2016)*, pp. 1–123, Oct 2017.

[22] D. Förster, F. Kargl, and H. Löhr, "PUCA: A pseudonym scheme with strong privacy guarantees for vehicular ad-hoc networks," *Ad Hoc Networks*, vol. 37, pp. 122–132, 2016.

[23] E. Verheul, "Activate later certificates for V2X: Combining ITS efficiency with privacy," Cryptology ePrint Archive 2016/1158, 2016. [Online]. Available: http://eprint.iacr.org/2016/1158

[24] J. Douceur, "The Sybil attack," in *Proc. of 1st Int. Workshop on Peer-to-Peer Systems (IPTPS).* Springer, 2002. [Online]. Available: www.microsoft.com/en-us/research/publication/the-sybil-attack/

[25] 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*, 2012, pp. 2036–2040.

[26] 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)*, Jan 2015, pp. 916–921.

[27] USDOT, "National security credential management system (SCMS) deployment support – SCMS baseline summary report," US Department of Transportation (USDOT), Tech. Rep., Jan 2018.

[28] Certicom, "Sec 4 v1.0: Elliptic curve Qu-Vanstone implicit certificate scheme (ECQV)," Certicom, Canada, Tech. Rep., 2013, http://www.secg.org/sec4-1.0.pdf.

[29] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[30] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) profile," Internet Engineering Task Force (IETF), Tech. Rep., 2008.

[31] E. Biham, "How to decrypt or even substitute DES-encrypted messages in 2^{28} steps," *Inf. Process. Lett.*, vol. 84, no. 3, pp. 117–124, nov 2002.

[32] N. Mouha and A. Luykx, "Multi-key security: The Even-Mansour construction revisited," in *Advances in Cryptology – CRYPTO 2015: 35th Annual Cryptology Conference.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 209–223.

[33] NIST, *Special Publication 800-131A Rev. 1: Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*, National Institute of Standards and Technology, Nov. 2015.

[34] F. Leighton and S. Micali, "Large provably fast and secure digital signature schemes based on secure hash functions," July 1995, US Patent 5,432,852.

[35] D. McGrew, M. Curcio, and S. Fluhrer, "RFC 8554 – Leighton-Micali hash-based signatures," IETF, Tech. Rep., apr 2019. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8554

[36] E. Andrade, M. Simplicio, P. Barreto, and P. Santos, "Lyra2: efficient password hashing with high security against time-memory trade-offs," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 3096–3108, 2016, See also: http://eprint.iacr.org/2015/136.

[37] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Eurocrypt'99.* Springer, 1999, pp. 223–238.

[38] M. Poumeyrol, M. Dahl, and M. Cornejo, "Paillier 0.2.0: A pure-rust implementation of the Paillier encryption scheme," https://docs.rs/crate/paillier/0.2.0, 2018.

[39] D. Bernstein, T. Lange, and P. Schwabe, "NaCl: Networking and cryptography library," https://nacl.cr.yp.to/index.html, 2016.

[40] D. J. Bernstein, "Curve25519: new diffie-hellman speed records," in *Int. Workshop on Public Key Cryptography.* Springer, 2006, pp. 207–228.

[41] VeriSign, "Total cost of ownership for public key infrastructure – white paper," www.imaginar.org/sites/ecommerce/index_archivos/guias/G_tco.pdf, Verisign, Inc., Tech. Rep., 2005.

[42] Entrust, "Why outsourcing your PKI provides the best value – white paper," www.entrust.com/wp-content/uploads/2013/05/Entrust-Managed-Services-PKI_TCO.pdf, Tech. Rep., July 2009.

[43] TCA, "Key decisions to progress Australian deployment of a security credential management system (SCMS)," Transport Certification Australia, Tech. Rep., Jan 2018. [Online]. Available: https://tca.gov.au/documents/Report-SCMS-TCA.pdf