ACPC: Efficient revocation of pseudonym certificates using activation codes

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

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

Abstract. Vehicular communication (V2X) technologies allow vehicles to exchange information about the road conditions and their own status, and thereby enhance transportation safety and efficiency. For broader deployment, however, such technologies are expected to address security and privacy concerns, preventing abuse by users and by the system's entities. In particular, the system is expected to enable the revocation of malicious vehicles, e.g., in case they send invalid information to their peers or to the roadside infrastructure; it should also prevent the system from being misused for tracking honest vehicles. Both features are enabled by Vehicular Public Key Infrastructure (VPKI) solutions such as Security Credential Management Systems (SCMS), one of the leading candidates for protecting V2X communication in the United States. Unfortunately, though, SCMS's original revocation mechanism can lead to large Certification Revocation Lists (CRLs), which in turn impacts the bandwidth usage and processing overhead of the system. In this article, we propose a novel design called Activation Codes for Pseudonym Certificates (ACPC), which can be integrated into SCMS to address this issue. Our proposal is based on activation codes, short bitstrings without which certificates previously issued to a vehicle cannot be used by the latter, which are periodically distributed to non-revoked vehicles using an efficient broadcast mechanism. As a result, the identifiers of the corresponding certificates do no need to remain on the CRL for a long time, reducing the CRLs' size and streamlining their distribution and verification of any vehicle's revocation status. Besides describing ACPC in detail, we also compare it to similar-purpose solutions such as Issue First Activate Later (IFAL) and Binary Hash Tree based Certificate Access Management (BCAM). This analysis shows that our proposal not only brings security improvements (e.g., in terms of resilience against colluding system authorities), but also leads to processing and bandwidth overheads that are orders of magnitude smaller than those observed in the state of the art.

Keywords: Vehicular communications · certificate revocation · activation codes · Security Credential Management System (SCMS) · Security · Privacy

1 Introduction

The longstanding pursuit for Intelligent Transportation Systems (ITS) [9] has been leading the automotive industry to expand the variety of computing and communication capabilities in vehicles, as well as in the roadside infrastructure. In particular, the increasing support to the so-called V2X communications technologies – including vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) mechanisms – allows the development of many ITS-oriented applications [15, 13]. Usually, such applications involve acquiring information about road conditions (e.g., its slope and current traffic load) and about the vehicles' state (e.g.,

velocity, acceleration, position and distance to other cars) [27]. Such information can then be shared among vehicles and also relayed to drivers, so adequate actions can then be taken manually or (semi-)automatically. The expected results include a reduction on traffic congestion, transportation delays, pollution emissions and waste of fuel, as well as increase safety for both vehicles and pedestrians [9, 7].

To become largely deployed and achieve their full potential, ITS technologies need to address some important concerns, in particular those related to security and privacy [30, 10]. Specifically, the authenticity of data exchanged via V2X is critical to prevent malicious users of abusing the system, such as forcing a target vehicle to stop by simulating an accident ahead. This can be accomplished via digital signatures computed over the broadcast messages, as well as with the revocation of a vehicle's certificates whenever misbehavior is detected. By relying on such Vehicular Public Key Infrastructure (VPKI), vehicles would only trust in, and act upon, authentic messages signed by non-revoked peers. However, to avoid privacy issues, the certificates employed by vehicles should not have the long lifespan typical of certificates traditionally employed on the Internet. Otherwise, the certificates themselves could be employed by eavesdroppers to track the mobility patterns of vehicles periodically broadcasting their positions. Actually, even when the broadcast messages do not contain a vehicle's exact position, tracking is still possible if the eavesdropper can pinpoint the origin of each message and the time in which it was sent.

A potential solution for these issues is the Security Credential Management System (SCMS) [34, 5], which is currently one of the leading VPKI candidate designs in the United States. Basically, SCMS proposes that each vehicle should carry multiple *pseudonym* certificates, i.e., certificates carrying no identifier of their owners. Even though each individual pseudonym certificate is short-lived, the batch of certificates carried by each vehicle is expected to cover a long time (e.g., a few years). As a result, messages signed by the same vehicle using different pseudonyms cannot be linked together, so privacy is preserved as long as its certificates are managed properly (e.g., a given certificate is not reused too often). Besides enabling the efficient generation of pseudonym certificates, SCMS also enables their revocation and linkage in case of misbehavior, thus facilitating investigations by law enforcement authorities whenever necessary. This revocation process is such that, with a single piece of information placed on a Certificate Revocation List (CRL), multiple certificates from a revoked vehicle can be identified as invalid.

One of the main advantages of SCMS's revocation procedure is that CRLs' size grows with the number of revoked vehicles, not with the number of certificates revoked. This is an important feature in the context of V2X due to the large number of pseudonym certificates carried by each vehicle. However, this benefit also comes with a drawback: a CRL entry's lifespan corresponds to the duration of the *batch* of pseudonym certificates carried by the revoked vehicle, meaning that those entries are not as short-lived as the certificates themselves. Since the batches are expected to cover 1-3 years [34], but this coverage may reach as much as 30 years [18], this is likely to lead to large CRLs, As a result, the bandwidth usage for the distribution of CRLs can become a burden if many vehicles are revoked. In addition, since SCMS's design is such that the cost of checking a certificate's revocation status is proportional to the number of entries in the CRL, the latter's growth also impacts the processing overhead at the vehicles.

Aiming to tackle this CRL expansion issue, in this article we improve SCMS's revocation process with the adoption of activation codes: small pieces of information without which pseudonym certificates previously issued become useless. Hence, by preventing revoked vehicles from obtaining those codes, the entries associated to those certificates can be safely removed from CRLs. The proposed design, named Activation Codes for Pseudonym Certificates (ACPC), builds upon previous works such as Issue First Activate Later (IFAL) [33] and Binary Hash Tree based Certificate Access Management (BCAM) [18]. Nevertheless, it addresses security and performance issues in both solutions, leading to a

more robust key revocation process for V2X communications.

The remainder of this document is organized as follows. Section 2 describes the SCMS architecture in further detail. Section 3 reviews related works aimed specifically at reducing CRL sizes in V2X communications. We then present the proposed ACPC solution is Section 4. Section 5 analyzes the security of ACPC's certificate issuance and revocation procedures. Section 6 compares our solution with related works, in terms of security and efficiency, from a theoretical and experimental perspective. Finally, Section 7 concludes the discussion.

1.1 General Notation

For convenience of the reader, Table 1 lists the main symbols and notation that appear along this document. Whenever pertinent, we assume standard algorithms for data encryption hashing and digital signatures. In particular, we assume that the following algorithms are employed: symmetric encryption (i.e., using shared keys) is done with the AES block cipher [23] whereas asymmetric encryption (i.e., involving public/private key pairs) is performed with ECIES [14]; hashing is performed with SHA-2 [25] or SHA-3 [26]; and the creation and verification of digital signatures uses algorithms such as ECDSA [24] or EdDSA [3, 16].

Table 1: General notation and symbols

Symbol	Meaning
G	The generator of an elliptic curve group
sig	A digital signature
$cert$	A digital certificate
$\mathrm{U}, \mathcal{U}$	Public signature keys (stylized $\mathcal{U}$: reserved for PCA)
u, u	Private keys corresponding to U and $\mathcal{U}$, respectively
S, s	Public and private caterpillar signature keys, respectively
E, e	Public and private caterpillar encryption keys, respectively
$\hat{S}, \hat{s}$	Public and private cocoon signature keys, respectively
$\hat{E}, \hat{e}$	Public and private cocoon encryption keys, respectively
f_1, f_2, f_a	Pseudorandom functions
V	Implicit certificate credential
la_id	ID of a Linkage Authority (LA)
cam_id	ID of a Certificate Access Manager (CAM)
$\mathtt{ls}_i$	Linkage seed
$\mathtt{plv}_i$	Pre-linkage value
$\mathtt{lv}$	Linkage value
β	Number of cocoon keys in pseudonym certificates batch
τ	Number of time periods in pseudonym certificate batch
σ	Number of certificates valid at any time period
α	Number of time periods covered by an activation period
`tree`	A binary hash tree
$\mathtt{node}(depth, count)$	The $count$-th node at level $depth$ in `tree`
$Hash(str)$	Hash of bitstring str
$Enc(\mathcal{K}, str)$	Encryption of bitstring str with key $\mathcal{K}$
$Dec(\mathcal{K}, str)$	Decryption of bitstring str with key $\mathcal{K}$
$Sign(\mathcal{K}, str)$	Signature of bitstring str, using key $\mathcal{K}$
$Verif(\mathcal{K}, str)$	Verification of signature on str, using key $\mathcal{K}$
$\lvert str \rvert$	The length of bitstring str, in bits.
$str_1 \,\|\| \, str_2$	Concatenation of bitstrings str_1 and str_2

2 The Security Credential Management System (SCMS)

The Security Credential Management System (SCMS) [34, 5] provides efficient mechanisms for the provisioning and revocation of pseudonym certificates in V2X communications. Combined, these mechanisms enable revocable privacy while ensuring that no single entity in the system can track honest vehicles. As a result, it addresses the needs of the "honest-but-curious" security model [17], in which the system's entities are expected to follow the correct protocols, but may engage in passive attacks such as trying to infer sensitive information from the exchanged data (e.g., aiming to track vehicles). Actually, it also takes into account attacks in a security model slightly more powerful than the "honest-but-curious" approach, which we call "dishonest-if-allowed": the system's entities may engage in active attacks, subverting the protocols' execution, but only if (1) this would bring them some advantage (e.g., the ability to track vehicles) and (2) if such misbehavior can go undetected.

After SMCS was published and became one of the leading candidate designs for protecting V2X security in the US [34, 5], different improvements to its design have been proposed [18, 32, 31]. Since the proposal hereby presented builds upon SCMS and can also take advantage of the aforementioned improvements, in what follows we describe SCMS in some detail, giving an overview of the state of the art and of the limitations targeted by our solution.

2.1 Overview

In SCMS, each vehicle receives two types of certificates: one enrollment certificate, which identifies authorized devices and is expected to have a long lifespan (e.g., years) before expiring; and multiple pseudonym certificates, each having a short validity (e.g., a few days), so $\sigma \geqslant 1$ certificates of this type are valid simultaneously. For protecting their privacy, vehicles may alternate the pseudonym certificates employed when sending different messages, thus avoiding tracking by nearby vehicles or by roadside units. In practice, however, it is important that σ remains small to limit the effects of "sybil-like" attacks [8], in which one vehicle poses as a platoon by sending multiple messages signed with different pseudonym certificates. Otherwise, a vehicle abusing its pseudonymity in this manner could, for example, receive preferential treatment from traffic lights programmed to give higher priority to congested roads [22].

As previously mentioned, SCMS was designed to allow the distribution of multiple pseudonym certificates to vehicles in an efficient manner, while providing mechanisms for easily revoking them if their owners misbehave. For this purpose, SCMS relies basically on the following entities (see Figure 1 for a complete architecture and [34] for the description of all of its elements):

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

- Misbehavior Authority (MA): identify misbehavior by devices and, whenever necessary, revokes them by placing their certificate identifiers into a CRL.

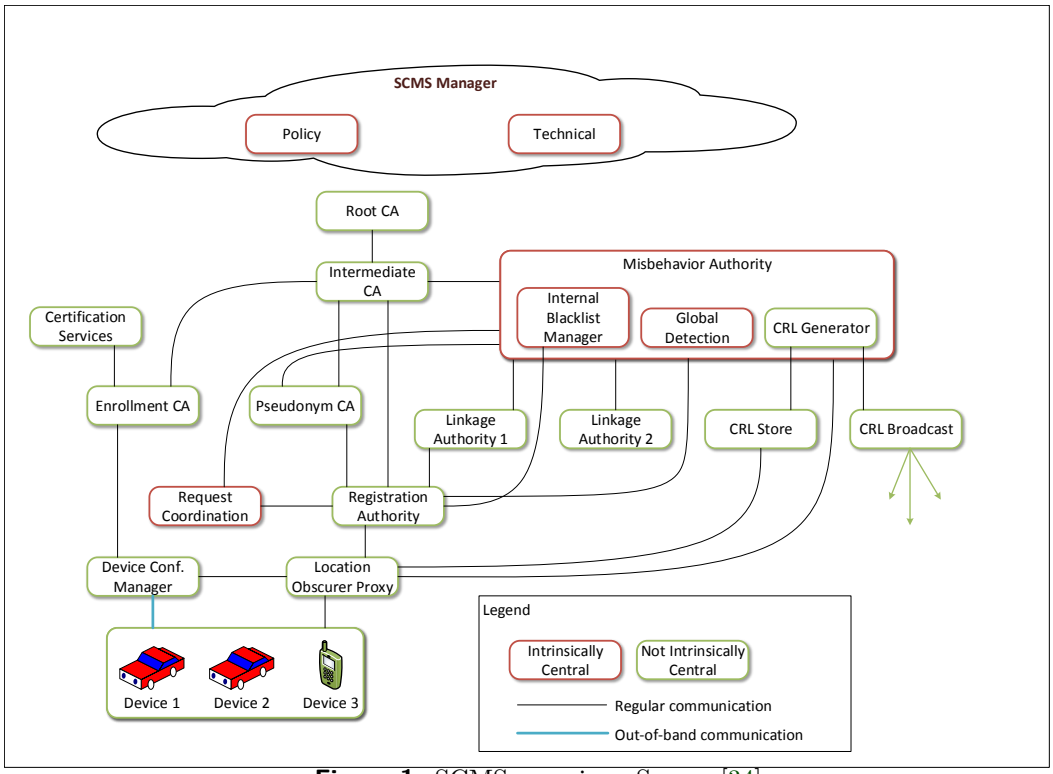

Figure 1: SCMS overview. Source:[34].

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

2.2 The butterfly key expansion

The butterfly key expansion enables vehicles to obtain arbitrarily large sets of (short-lived) pseudonym certificates by means of a single, small-sized request message from vehicles. Figure 2 illustrates this process, whereas Table 2 lists the corresponding operations, explained in what follows.

The vehicle starts by picking two random *caterpillar* private keys, s and e, from which the corresponding public caterpillar keys $S = s \cdot G$ and $E = e \cdot G$ are computed. It also picks two random seeds for initializing pseudorandom functions f_1 and f_2. The quadruple (S, f_1, E, f_2) is then sent to the Registration Authority (RA).

The RA then generates β public *cocoon* signature keys $\hat{S}_i = S + f_1(i) \cdot G$, where $0 \leqslant i < \beta$. Analogously, the RA uses E for generating β public cocoon encryption keys $\hat{E}_i = E + f_2(i) \cdot G$. Pairs of public cocoon keys $(\hat{S}_i, \hat{E}_i)$ from different vehicles are then shuffled together, and sent individually to the Pseudonym Certificate Authority (PCA) for the generation of the corresponding pseudonym certificates.

The subsequent procedure followed by the PCA when processing $(\hat{S}_i, \hat{E}_i)$ depends on whether explicit or implicit certificates [6] are employed. For explicit certificates, the PCA picks a random r_i and computes the vehicle's public signature key as $\mathrm{U}_i = \hat{S}_i + r_i \cdot G$. The PCA then digitally signs this public key together with any required metadata `meta`

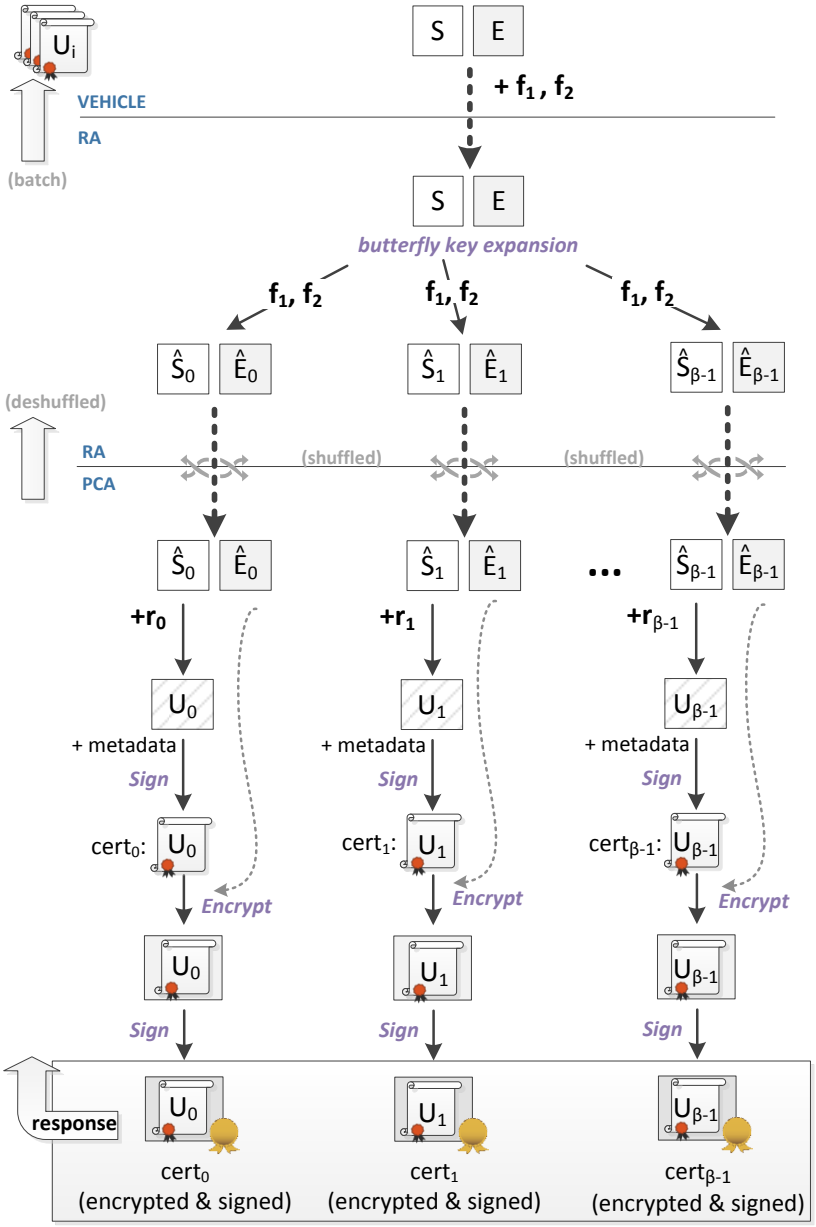

Figure 2: SCMS's butterfly key expansion and pseudonym certificate generation.

(e.g., that key's validity period), obtaining the signature sig_i, so the resulting explicit pseudonym certificate $cert_i$ is given by the triple $(\mathrm{U}_i, \texttt{meta}, sig_i)$. For implicit certificates, this process is slightly different: the PCA starts by computing a credential $V_i = \hat{S}_i + r_i \cdot G$, once again for a random r_i, so the implicit certificate $cert_i$ is defined as the pair $(V_i, \texttt{meta})$. The PCA then signs this certificate to obtain $sig_i = h_i \cdot r_i + u$, where $h_i = Hash(cert_i)$. Whichever the certification model adopted, the resulting certificate and its companion data (namely, r_i for explicit certificates, and sig_i for implicit ones) is encrypted with $\hat{E}_i$. To avoid Man-in-the-Middle attacks by the RA, this encrypted package is also signed with the PCA's own private key u. The PCA's response is then sent to the RA, which gathers and de-shuffles them before relaying the corresponding certificates to the requesting vehicle.

Finally, the vehicle verifies the PCA's signature on the encrypted certificates, uses the

Table 2: The butterfly key expansion process for issuing pseudonym certificates.

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

private cocoon encryption key $\hat{e}_i = e + f_2(i)$ to decrypt the PCA's response, and verifies the validity of the pseudonym certificate thereby enclosed. For explicit certificates, this verification consists in checking that the certificate's signature is valid and that the public key U_i was indeed derived from s and r_i, i.e. that $\mathrm{U}_i = (s + f_1(i) + r_i) \cdot G$; in that case, the vehicle's private key is set to $\mathrm{u}_i = s + f_1(i) + r_i$. When the certificates are implicit, the vehicle simply follows the regular implicit verification protocol described in [6]: it computes its own private key as $\mathrm{u}_i = h_i \cdot (s + f_1(i)) + sig_i$, sets the corresponding public key to $\mathrm{U}_i = \mathrm{u}_i \cdot G$, and verifies that $\mathrm{U}_i = h_i \cdot V_i + \mathcal{U}$, where $\mathcal{U}$ is the PCA's public signature key.

Independently of the type of certificate adopted, the vehicles' privacy is protected in this process as long as there is no collusion between RA and PCA. Specifically, the shuffling of public cocoon keys performed by the RA prevents the PCA from learning whether or not a group of keys received belong to a same device. Unlinkability of public keys towards the RA, in turn, is obtained because the latter does not learn the value of $cert_i$ in the PCA's encrypted response.

2.3 Key linkage

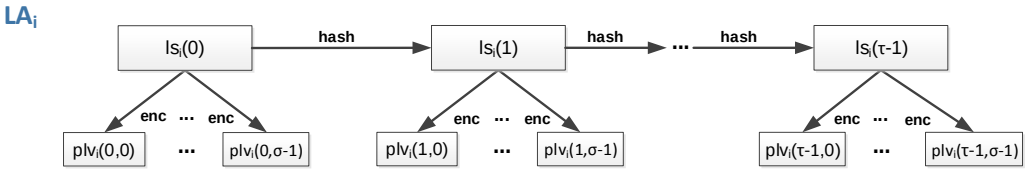

Figure 3: SCMS's key linkage process: generation, by LA_i, of pre-linkage values employed for certificate revocation.

To avoid large certificate revocation lists (CRLs), revocation in SCMS is done in such a manner that many certificates from a same user can be linked together by inserting only a small amount of information into a CRL. For this purpose, each pseudonym certificate generated by the PCA includes a linkage value $\texttt{lv}$, computed by XORing two pre-linkage values, $\texttt{plv}_1$ and $\texttt{plv}_1$, provided by two different Linkage Authorities (LA). The generation of $\texttt{plv}_i$ by LA_i is done upon request by the RA, as follows (see Figure 3).

First, LA_i picks a random, 128-bit linkage seed $\texttt{ls}_i(0)$. Next, if the RA's request covers τ certificate time periods, LA_i iteratively computes a τ-long hash chain [19] $\texttt{ls}_i(t) = Hash(la\_id_i \,\|\, \texttt{ls}_i(t-1))$, where la_id_i is LA_i's identity string and $1 \leqslant t < \tau$. Each $\texttt{ls}_i(t)$

is then used in the computation of σ pre-linkage values $\texttt{plv}_i(t,c) = Enc(\texttt{ls}_i(t), la\_id_i \,\|\, c)$, for $0 \leqslant c < \sigma$. Finally, every $\texttt{plv}_i(t,c)$ is truncated to a suitable length, individually encrypted and authenticated using a key shared between the PCA and LA_i, and then sent to the RA. The RA simply includes this encrypted information, together with the corresponding cocoon keys, in the requests sent to the PCA. As a result, the latter can compute the linkage value to be included in the the c-th certificate valid in time period t as $\texttt{lv}(t,c) = \texttt{plv}_1(t,c) \oplus \texttt{plv}_2(t,c)$.

Whenever the Misbehavior Authority (MA) identifies a malicious vehicle, its non-expired certificates can be revoked altogether. This is accomplished via the collaboration among PCA, RA, and LAs. Namely, the PCA can associate the `lv` informed by the MA to the original pseudonym certificate request received from the RA. The PCA then provides this information, together with the corresponding pre-linkage values $\texttt{plv}_i(t,c)$, to the RA. The RA, in turn, can (1) identify the vehicle/driver behind that certificate request, placing the corresponding enrollment certificate in a blacklist for preventing its owner from obtaining new pseudonym certificates, and (2) ask LA_i to identify the linkage seed $\texttt{ls}_i(0)$ from which $\texttt{plv}_i(t,c)$ was computed. Finally, each LA_i provides RA with $\texttt{ls}_i(t_s)$, where t_s is the time period from which the revocation starts being valid (e.g., the time period when the misbehavior first occurred, or the current one). The set of $\texttt{ls}_i(t_s)$ received from the LAs can then be placed in a CRL to be distributed throughout the system, allowing any entity to compute $\texttt{lv}(t,c)$ for time periods $t \geqslant t_s$, linking the corresponding certificates to a single CRL entry. Consequently, the misbehaving vehicle's *future* certificates are revoked and can be linked together; *past* certificates remain protected, though, preserving the vehicle's privacy prior to the detection of the malicious activity.

In terms of computational costs, this revocation process is such that each revoked device contributes with 2 pre-linkage values to the CRL. Hence, the CRL grows linearly with the number of revoked vehicles, not with the number of revoked certificates. The main drawback of this gain in size is that checking whether a given certificate is in the CRL requires the verification of *every* CRL entry against that certificate's linkage value. More precisely, for each CRL entry published at time period t_s, the verification of whether it covers a given certificate involves basically the computation of two components:

a) $\texttt{ls}_i(t_c)$: it takes $2 \cdot (t_c - t_s)$ hashes to compute $\texttt{ls}_i(t_c)$ from $\texttt{ls}_i(t_s)$, where $i = \{1,2\}$ and t_c is the time period when the verification is performed. This cost may be reduced by means of pre-computation, i.e., if the vehicles always keeps the updated version of the linkage seeds, $\texttt{ls}_i(t_c)$, besides the original ones provided in the CRL. Nevertheless, to cope with the lack of a system-wide time synchronization [33], vehicles may actually need to keep a slightly older linkage seed in memory; for example, by keeping $\texttt{ls}_i(t_c - \epsilon)$ for a small ϵ, it is possible to compute $\texttt{ls}_i(t_c)$ with only ϵ hashes.

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

3 Related works

As discussed in Section 2.3, the cost of checking a certificate's revocation status given a CRL is linearly dependent on the number of entries in that CRL. Therefore, keeping this number low is important not only to save bandwidth when distributing CRLs, but also to allow a faster and more energy-efficient verification of a certificate's revocation status. Unfortunately, however, the same mechanism proposed in SCMS for shortening CRLs, which associates several certificates to a same entry, also extends the lifetime of

those entries: after all, linkage seeds placed into a CRL can only be safely removed from it after all certificates associated to those seeds have expired. Consequently, even if device revocation events occur at a low frequency, CRLs may actually grow big because the corresponding entries remain in the CRL for a duration comparable to that of certificate batches (e.g., years). Actually, this issue is present even in recently proposed improvements to SCMS's key-revocation process. For example, in [5], the pre-linkage values are computed using a Davies-Meyer construction [28] aiming to avoid possible vulnerabilities related to the reversibility of encryption, a modification that does not affect the longevity of the CRLs' entries. As another example, in [31] the linkage and pre-linkage values are computed with suffix-free hashes, which improves the system's resilience against birthday attacks, and another layer in the hash tree is added to simplify the temporary linkage/revocation of vehicles. Even though entries for temporarily revoked vehicles can be swiftly removed from CRLs, this facility does not apply to entries related to permanent revocations.

One possible approach for reducing the size of CRLs is to rely on Bloom Filters [29, 11]. More precisely, this probabilistic data structure allows the CRL list to be compressed, although this comes at the cost of false-positives (but no false-negatives) on the pertinence test. Although potentially useful for reducing CRL sizes when many certificates need to be revoked, this approach is likely to be inefficient when applied to the revocation of only a few vehicles. The reason is that the filter's size depends on the maximum number of certificate revocations supported, so the CRL always maintains its full size despite the actual number of revoked certificates. Given the large number of pseudonym certificates per vehicle, Bloom filters are potentially more efficient than placing in the CRL every single certificate belonging to a revoked vehicle. However, such technique is not as appealing when combined with solutions such as SCMS, in which all certificates from a single vehicle can be revoked altogether with a single CRL entry. In this scenario, Bloom filters are useful mostly to store multiple certificate identifiers at the vehicles' side, as proposed in [12], which makes this solution complementary to activation codes.

A more promising approach, which is also adopted in ACPC, is to prevent a revoked vehicle from accessing its pseudonym certificates. This is the basic idea behind the Issue First Activate Later (IFAL) scheme [33]. In IFAL, non-revoked vehicles can periodically query the system for the "activation codes", small pieces of information required to compute the private keys for their own certificates. As a result, the identifiers for revoked certificates that have been activated need to remain in the CRL only until their corresponding expiration dates, since subsequent certificates issue to the same vehicle cannot be activated. Albeit interesting, IFAL suffers from a significant security issue: since the PCA communicates directly with the vehicles, it can link the pseudonym certificates it issues to the corresponding vehicle's enrollment certificates, even without colluding with any other entity. The only protection offered by IFAL against such linkability risk is that it requires the PCA to delete the information produced during the certificate issuance process [33, Sec. 3.2]; however, such requirement can be easily ignored by a dishonest PCA. Unfortunately, this makes the IFAL less secure than SCMS itself, hindering an eventual integration between the two solutions.

The Binary Hash Tree based Certificate Access Management (BCAM) scheme [18] is another scheme that uses activation codes for reducing CRL sizes. Unlike IFAL, however, BCAM was designed to interoperate with the SCMS architecture, inheriting its ability to protect the privacy of honest users against a dishonest PCA as long as it does not collude with other system entities. More precisely, in BCAM the RA does not simply relay the PCA-encrypted certificate batches to the requesting vehicle, but first sends them to a Certificate Access Manager (CAM), which encrypts those batches using a symmetric cipher. The CAM-encrypted batches can only be decrypted by vehicles that are able to compute corresponding activation codes, called device specific values (DSVs). Instead of requiring vehicles to explicitly request their DSVs, though, the CAM builds a binary hash

tree whose leaves allow the DSVs to be computed, and then broadcasts the nodes of the tree that allow non-revoked vehicles (and only them) to recover their own DSVs. As a result, the distribution of activation codes in BCAM is considerably more efficient that the one in IFAL. Nevertheless, and even though BCAM does improve SCMS's revocation process, it can be further improved in terms of security and efficiency. Namely, one security drawback of BCAM's design is that it creates an extra point of collusion in the SCMS architecture: the CAM, like the RA, learns which batch of PCA-encrypted certificates belong to a same vehicle; consequently, the CAM can collude with the PCA to violate those certificates' unlinkability and, hence, the users' privacy. In addition, during the butterfly key expansion, the CAM encrypts certificates that have already been encrypted by the PCA; therefore, this process can be further optimized if those two encryption processes are combined into one. Both issues are addressed in the proposed ACPC solution.

4 Proposal: Activation Codes for Pseudonym Certificates (ACPC)

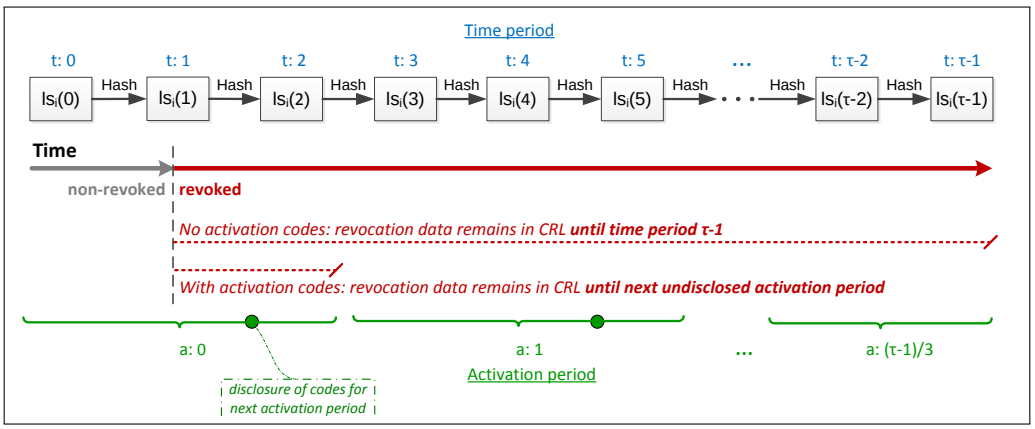

Figure 4: Activation codes and permanence of revocation data (e.g., linkage seeds) in CRLs.

To avoid the growth of CRLs while preserving the performance gains associated with the butterfly key derivation, we build upon the concept of *activation codes*: bitstrings without which certificates previously acquired cannot be used (namely, in the proposed solution, they cannot be decrypted). Each activation code enables the usage of certificates corresponding to a certain *activation period*, which spans $\alpha \geqslant 1$ certificate time periods. This is illustrated in Figure 4, for activation periods covering $\alpha = 3$ time periods.

Those codes are then periodically disclosed to non-revoked vehicles before the start of the corresponding time periods to allow a timely activation of their own certificates. Revoked vehicles are prevented from obtaining activation codes for their certificates, at least until this revocation status is removed. As a result, identifiers of revoked certificates that cannot be activated do not need to remain in CRLs, reducing the sizes of the latter. For example, certificates could be valid for 1 week, whereas the activation period could be set to 3 weeks and be disclosed 1 week before they are actually required. In this case, identifiers for certificates from revoked vehicles would have to remain in CRLs for at most 4 weeks: if the codes for the next activation period have not yet been disclosed, then the CRL needs to cover only the current activation period (at most 3 weeks); otherwise, the CRL needs to cover the next activation period (3 weeks) and the remainder of the current one (at most 1 week). After that, revoked devices do not receive new activation codes.

The proposed solution, named ACPC (Activation Codes for Pseudonym Certificates), is inspired by works such as IFAL [33] and BCAM [18]. However, it innovates by addressing the main shortcomings of these solutions in terms of performance and security, as further discussed in Section 6.

4.1 Generating activation codes: binary hash trees

Similarly to BCAM, we assume the existence of one or more Certificate Access Managers (CAM), each having a different identifier *cam_id*. CAMs are entities responsible for creating and distributing activation codes. For this purpose, a CAM creates a binary hash tree $\texttt{tree}^t$ for each time period t, using a pre-image resistant hash function, as illustrated in Figure 5. If the activation period spans $n+1$ time periods, then $\texttt{tree}^t = \ldots = \texttt{tree}^{t+n}$.

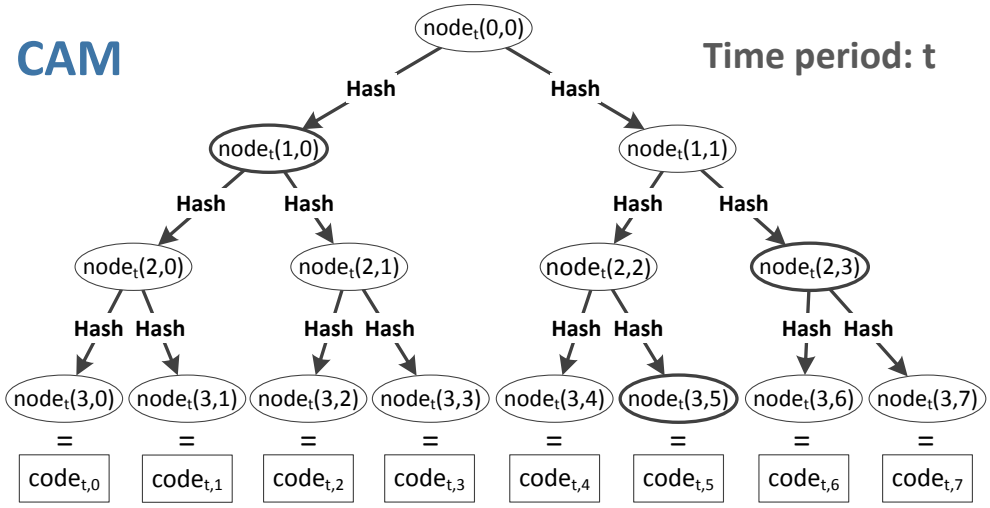

Figure 5: Activation tree generated by CAM. The activation codes correspond to the leaves of the binary hash tree.

The tree's nodes are denoted $\texttt{node}^t(depth, count)$, where $depth \geqslant 0$ and $0 \leqslant count \leqslant 2^{depth} - 1$ indicate the node's position inside the tree. The height of the tree must match the length of the vehicles' identifiers (VID), in bits. As a result, each leaf $\texttt{node}^t(depth, count)$ can be used to represent a single vehicle in the system: the one with VID = *count*. In [18], for example, the suggested length of VID is 40 bits, which is enough to cover more than 1 trillion vehicles. For brevity of notation, we denote by $\texttt{code}_{t,\text{VID}}$ the leaf of $\texttt{tree}^t$ whose index corresponds to a given VID, i.e., $\texttt{code}_{t,\text{VID}} = \texttt{node}^t(|\text{VID}|, \text{VID})$.

The nodes of $\texttt{tree}^t$ are assumed to be k-bit long, yielding a k-bit security level (e.g., in modern deployments one would expect $k = 128$). The tree is built in the following manner. First, its root $\texttt{node}^t(0,0)$ is set to a (pseudo)random bitstring, unique for each activation period. Every other node is then computed from its parent node combined with a "security string" I, i.e., a different suffix for each node. More precisely, we have:

$$\texttt{node}^t(depth, count) = Hash(\texttt{node}^t(depth - 1, \lfloor count/2 \rfloor) \,\|\, I)$$

where the security string I is defined as

$$I = (cam\_id \,\|\, t \,\|\, depth \,\|\, count)$$

If the activation period spans multiple time periods, then t is set to the first time period covered by that activation period. This approach gives the system enough flexibility to

increase or reduce the length of the activation periods without incurring the repetition of security strings. As further discussed in A, such non-repeatable security strings are useful to thwart birthday attacks analogous to those described in [4, 31].

Table 3 shows suggested lengths for the fields that compose those security strings in ACPC, leading to $|I| = 104$. This composition is designed to support 40-bit long VIDs for 2^{24} time periods, which means more than 300.000 years if the time periods are 1 week long. At the same time, it is unlikely to have any perceptible impact on the computation of activation trees, 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) [25]; therefore, a single call to its underlying compression function is enough to process a 128-bit node value value even when it is combined with a 319-bit or smaller security string.

Table 3: Components of the security strings employed in ACPC's activation trees.

Field	Suggested length (bits)	Description
depth	8	Node's depth in tree, starting at 0. Mandatory: $\|depth\| \geqslant \lg(\|\text{VID}\|)$.
count	40	Node's index in the depth, starting at 0. Mandatory: $\|count\| \geqslant \|\text{VID}\|$.
t	24	Time period to which the tree is associated
cam_id	32	CAM's identifier

4.2 Issuing pseudonym certificates with activation codes

Figure 6 illustrates ACPC's pseudonym certificate issuance, in which the binary hash tree generated by the CAM is integrated into SCMS's butterfly key expansion process. As shown in this figure, when a vehicle with a given VID requests a batch of pseudonym certificates, it still provides the quadruple (S, f_1, E, f_2) to the RA as described in Section 2.2. This should then lead to the generation of $\beta = \tau \cdot \sigma$ certificates, so a total of σ certificates are valid for each of the τ time periods covered in the batch.

Upon reception of the vehicle's request, the RA queries the CAM for the blinded activation codes $A_t = f_a(\texttt{code}_{t,\text{VID}}, t, \text{VID}) \cdot G$. This operation uses a pseudorandom function f_a that is public, i.e., whose seed does not need to be kept secret. Nevertheless, its output at this point is unpredictable: after all, it involves the activation tree's leaf $\texttt{code}_{t,\text{VID}}$, which has not yet been disclosed by the CAM. In addition, f_a's output is blinded by the CAM via the multiplication by the elliptic curve generator G, so the actual activation codes cannot be learned by the RA from the CAM's response.

The RA then proceeds with the butterfly key expansion process in a manner that is very similar to the process described in Section 2.2, except for one important difference: the blinded codes provided by the CAM are added to the computation of the public cocoon encryption keys. More precisely, let $\hat{E}_{t,c}$ denote the c-th public cocoon encryption key that belongs to time period t. Each of those keys is then computed by the RA as $\hat{E}_{t,c} = E + A_t + f_2(t \cdot \sigma + c) \cdot G$, where $0 \leqslant c < \sigma$ and $0 \leqslant t < \tau$. As usual, the resulting cocoon keys are shuffled together with keys from other vehicles, before being sent to the PCA. By using f_2 in the computation of cocoon encryption keys, the RA ensures that they cannot be later correlated by the CAM or by the PCA, even for groups of keys computed using the same A_t. Therefore, this process preserves the unlinkability of pseudonym certificate requests, whether or not there is a collusion between CAM and PCA.

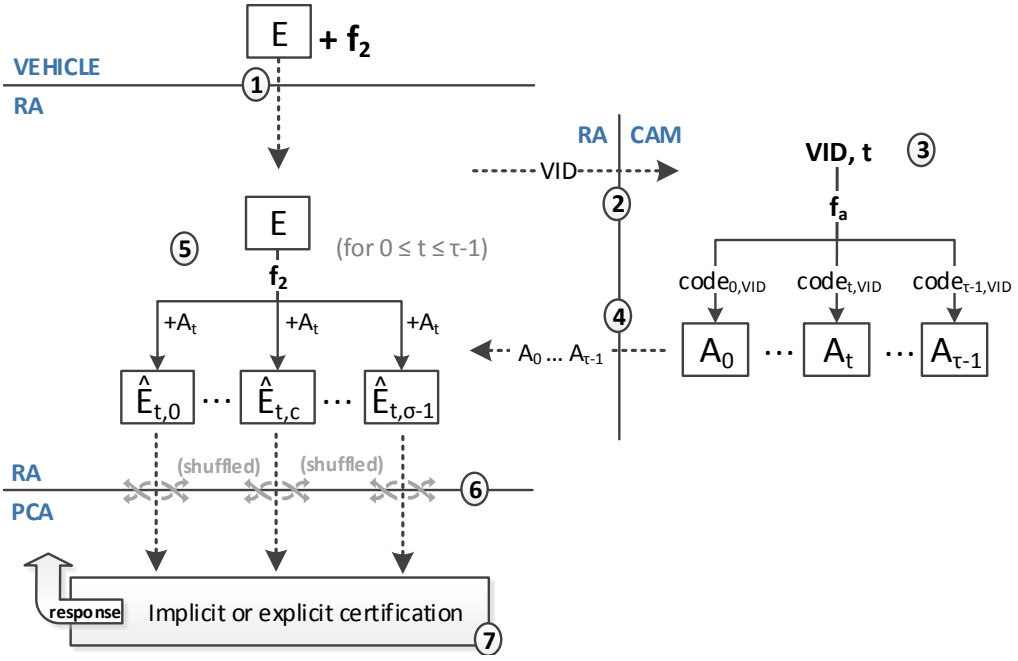

Figure 6: Issuing pseudonym certificates with activation codes. For simplicity, we assume that each activation period covers a single time period.

Finally, the PCA computes the vehicle's (implicit or explicit) pseudonym certificate, which is encrypted with $\hat{E}_{t,c}$ before being sent back to the RA. The RA, in turn, simply relays those certificates to the corresponding vehicle, without contacting the CAM once again. Since this procedure is identical to the one proposed in SCMS, the processing costs and bandwidth usage at the PCA remain unchanged. In addition, the underlying security properties discussed in Section 2.2 still apply, including the protection against MitM attacks performed by the RA and the unlinkability among certificates unless PCA and RA collude.

The result of this process is that the encrypted certificates obtained by the vehicle can only be decrypted if the associated $\texttt{code}_{t,\text{VID}}$ is also obtained. After all, the decryption key corresponding to $\hat{E}_{t,c}$ is now computed as $\hat{e}_{t,c} = e + f_a(\texttt{code}_{t,\text{VID}}, \text{VID}, t) + f_2(t{\cdot}\sigma + c)$. Therefore, to keep a vehicle with identifier VID_r from activating its own certificates, it suffices to prevent it from obtaining $\texttt{code}_{t,VID_r}$. In that case, entries for that vehicles' certificates do not need to remain in CRLs for too long.

As a final remark, we note that the activation codes as hereby described can also combined with the optimized, unified butterfly key expansion process proposed in [32]. More precisely, this alternate design achieves better performance by replacing the pair of caterpillar keys (S, E) with a a single key $X = x \cdot G$. Therefore, since the corresponding public cocoon keys $\widehat{X}$ are employed by PCA for encrypting its response, the RA simply needs to include the blinded activation codes in their computation by making $\widehat{X}_{t,c} = X + A_t + f_1(t{\cdot}\sigma + c) \cdot G$.

4.3 Distributing activation codes for non-revoked devices

In ACPC, like in BCAM, activation codes can be broadcast by the CAM to all non-revoked vehicles rather than individually sent to them upon request. This eliminates the need for bidirectional connectivity between vehicles and CAMs when distributing activation codes.

The tree's nodes that must be broadcast depend on which nodes are currently re-

voked/suspended, taking into account that every node of a binary hash tree can be computed from its parent. For example, given the root of the tree, all of its leaves can be computed and, thus, all vehicles can obtain their corresponding activation codes. Hence, if no vehicle is revoked in time period t, the CAM only needs to broadcast $\texttt{node}^t(0,0)$ to allow all certificates in the system to be activated. This leads to optimal performance when distributing activation codes.

When a vehicle needs to be revoked, however, the CAM must not reveal any of the nodes in the path between the corresponding leaf and the tree's root. This prevents the computation of that leaf not only by the revoked vehicle, but also by non-revoked vehicle that might try to collude with revoked ones to allow the activation of the latter's certificates. For example, consider the tree shown in Figure 5. To revoke the vehicle whose VID is 4, the CAM would have to broadcast only the following nodes (shown with ticker borders in that figure): $\texttt{node}(1,0)$, which enables the computation of leaves $\texttt{node}(3,0)$ through $\texttt{node}(3,3)$; $\texttt{node}(2,3)$, used to compute leaves $\texttt{node}(3,6)$ and $\texttt{node}(3,7)$; and the leaf $\texttt{node}(3,5)$. More generally, and as mentioned in [18], when n_r users out of n_t are revoked, the number of nodes included in the message broadcast by the CAM is on average $n_r \cdot \lg(n_t/n_r)$ for $1 \leqslant n_r \leqslant n_t/2$ (cf. Theorem 1 of [1]). Hence, albeit more expensive than the scenario in which no revocation occurs, this approach scales quite well, and is still more efficient than the individual delivery of each activation code. Actually, there are also efficient methods for encoding binary hash trees such as those hereby described, so the index of each node included in the broadcast message can be represented with less than $|\text{VID}|$ bits (cf. Section 4.4 of [18]), saving some bandwidth.

5 Security analysis

In this section, we discuss the security of ACPC's certificate issuance and revocation procedures.

5.1 Security of the certificate issuance process

In ACPC, a collusion between CAM and PCA (resp. RA) reveals as much information as the PCA (resp. RA) had available in the original SCMS. Indeed, if we remove the influence of A_t over the public cocoon keys computed as described in Section 4.2, the result matches the public cocoon keys in SCMS. Therefore, a collusion with the CAM can only remove the entropy introduced by this entity, while still preserving the security properties discussed in Section 2.2.

In particular, a vehicle's private caterpillar key x remains protected by the elliptic curve discrete logarithm problem (ECDLP) during the whole execution of the protocol. Hence, RA, PCA and CAM are unable to recover the signature or decryption private keys derived from it, even if they collude. Unlinkability among certificates is similarly preserved, as long as the RA and PCA do not collude: the shuffling done by the RA still hides from the PCA any relationship between certificate requests intended for a same vehicle; meanwhile, the PCA's encrypted response prevents anyone but the appropriate vehicle from learning $cert_i$. Finally, since the butterfly key expansion process grants vehicles the ability to verify whether or not the received certificates were generated in a fair manner, MitM attacks (e.g., by the RA) are averted.

It is worth mentioning that the aforementioned CAM's inability to create a new threat via collusion with other entities is not just a fortunate coincidence. Actually, the (unblinded) activation codes $\texttt{code}_{t,\text{VID}}$ are the only information initially kept secret by the CAM and, thus, that could be contributed in such collusion. Since those codes are periodically disclosed to allow vehicles to activate their certificates, though, by design

such public disclosure should not negatively impact the system's security. Consequently, a "private disclosure" during a collusion is expected to have an equivalent result.

5.2 Security of the revocation procedure

The security of ACPC's revocation procedure relies on the first pre-image resistance of the hash function employed for the construction of activation trees, as well as the proper disclosure of its nodes by the CAM. In principle, following a dishonest-if-allowed model, this means that the choice of a secure hash function is enough to enforce revocation. At least this should be the case if we assume that the system's entities would gain nothing by illegitimately un-revoking vehicles, i.e., without authorization from the MA. Nonetheless, it is useful to evaluate what happens when one of the system's entities is compromised and, as a result, its capabilities are misused aiming to allow the activation of revoked devices.

On the one hand, a rogue/compromised CAM could disclose the roots of every activation tree to all vehicles, even revoked ones, allowing all certificates in the system to be activated. This would not give the attacker any advantage, though, besides disrupting the system's ability to revoke devices in an efficient manner. In particular, in consonance with the discussion in Section 5.1, this would not grant the CAM or any other entity the ability to track devices. Consequently, in a dishonest-if-allowed scenario, it is unlikely that the CAM itself would go rogue and engage in such malicious activity. Furthermore, if the CAM's storage is somehow compromised, the leakage of codes can still be contained by keeping the revoked vehicles' data in CRLs, just like in the original SCMS. Hence, the attack would not result in any catastrophic security breach, but only nullify the performance gains provided by activation codes.

On the other hand, a security breach at the RA or PCA should not reveal any information about activation codes. The reason is that these entities never learn those codes, which are only known by the CAM. Nevertheless, if any of these entities goes rogue or is compromised at a level that allows its behavior to be controlled by attackers, it can provide valid certificates to revoked vehicles independently of activation codes. Specifically, a dishonest PCA can always issue new pseudonym certificates for vehicles, including revoked ones, at least until the PCA itself is revoked. A compromised RA could act similarly, e.g., by requesting pseudonym certificates for a non-revoked VID_d, and then delivering those certificates to a revoked vehicle whose identifier is $VID_r \neq VID_d$. Such misbehavior is likely to go unnoticed because the certificates do not carry any VID on them. In addition, if VID_d corresponds to a valid vehicle whose pseudonym certificates have not been requested yet, the CAM would not be able to notice the fraud by the RA: from the CAM's perspective, this is simply a new vehicle in the system.

Actually, even if VID_d has already been requested in the past, it would be difficult for the CAM to use this knowledge to prevent such attack. The reason is that, if the CAM is configured to refuse a second request for the same VID_d, management issues are likely to arise. For example, the processing of the initial request for VID_d may naturally fail, so an honest RA would actually need to send a second request referring to the same VID_d. As another example, a dishonest RA might abuse this process by performing a "denial-of-certification" attack: the RA queries the CAM requesting the caterpillar keys for a non-revoked VID_d, but it does not execute the pseudonym certificate issuance process; future requests referring to VID_d, potentially by honest RAs, would then fail.

These observations indicate that, even if a rogue RA or PCA never gains access to activation codes, their roles in the system still enable them to provide valid certificates for revoked vehicles. Actually, a similar discussion also applies to the original BCAM protocol, in which a rogue RA or PCA could provision revoked vehicles with the PCA-encrypted certificates, before they are once again encrypted by the CAM. Even though ACPC's approach of ensuring that only the CAM is able to distribute activation codes does not actually prevent such threats, it was adopted because it does reduce the system's attack

surface. For example, suppose that RA and/or PCA store the (PCA-encrypted) certificates generated for the purposes of disaster recovery, as proposed in [18, Section 5.3.3]. In this case, a data breach disclosing the certificates issued for a vehicle that is now revoked does not create any security concern, since that vehicle remains unable to decrypt those certificates.

6 Comparison with related works: IFAL and BCAM

When compared to IFAL, ACPC differs in at least two important aspects. First, IFAL allows an "honest but curious" PCA to link several certificates to a same device; this privacy issue is absent in the solution hereby described if we assume, like in the original SCMS, that PCA and RA do not collude. Second, ACPC allows vehicles to obtain activation much more efficiently than IFAL's strategy, using binary hash trees to broadcast activation codes rather than requiring vehicles to individually request them.

Architecturally, ACPC shares more similarities with BCAM than with IFAL, in particular because both BCAM and our proposal use binary hash trees for the distribution of activation codes. However, ACPC provides better security, for at least two reasons. First and foremost, the fact that the CAM does not receive certificates from the RA prevents the former from learning which PCA-encrypted certificates belong to a same device; therefore, and unlike BCAM, a collusion between CAM and PCA does not allow those entities to track vehicles. Second, as discussed at the end of Section 5.2, ACPC also reduces the revocation procedure's attack surface by a rogue RA or PCA, so it is able to protect the vehicle's privacy even if PCA and CAM collude or are compromised.

As another relevant benefit, ACPC is more efficient than BCAM in terms of processing and bandwidth usage, as discussed in the following subsections.

6.1 Bandwidth usage

By integrating security strings into the activation trees, the nodes of those trees can be 128-bit long while still preserving a 128-bit security level, despite the number of revoked devices. When compared to BCAM, which uses 256-bit nodes for achieving the same security level (see A), this represents a 50% bandwidth gain for the distribution of activation trees.

In addition, ACPC also saves bandwidth between RA and CAM during the pseudonym certificate issuance process, because these entities only exchange data that represent nodes from activation trees, rather than batches of certificates. More precisely, assume that SCMS adopts implicit certificates, which are usually shorter that explicit ones, and employs the optimizations described in [32], which removes the need of signing the encrypted package pkg. In that case, each PCA-encrypted response $pkg = Enc(\hat{E}_i, \{cert_i, sig_i\})$ is expected to be such that $|pkg| \geqslant 6k$ bits for a security level of k: (1) $2k$ bits for the implicit credential V_i enclosed in $cert_i$, since it corresponds to an elliptic curve point; (2) $2k$ bits for the random elliptic curve point that is generated during any ECIES-based encryption; (3) $2k$ bits for the signature sig_i; and (4) some extra bits for the certificate's metadata (e.g., a linkage value and an expiration date), as well as for the authentication tag resulting from the ECIES-based encryption, which we ignore in our calculations. Hence, for a batch size $\beta = \tau \cdot \sigma$, in BCAM the total bandwidth usage at the RA and CAM would be $2 \cdot (6k) \cdot (\tau \cdot \sigma)$ bits, where the "2" multiplication factor is due to the fact that this cost is paid both downstream and upstream: the RA sends PCA-encrypted batches to the CAM, and then receives CAM-encrypted batches.

In comparison, in ACPC the CAM simply sends τ/α blinded activation codes to the RA, where $\alpha \geqslant 1$ is the number of time periods covered by each activation period. Since each blinded activation code A is an elliptic curve point (i.e., $2k$ bits), this corresponds to

an upstream (resp. downstream) cost at the CAM (resp. RA) of $2k \cdot \tau/\alpha$ bits. Therefore, ACPC is expected to use approximately $1/(6\sigma \cdot \alpha)$ of the bandwidth required by BCAM during the pseudonym certificate issuance process. To give a numeric example, suppose that the system is configured in such a manner that 20 certificates are valid per time period (as in [18]), and each activation period covers 4 time periods (e.g., for certificates valid for a week, the activation period would correspond to a month). In this case, issuing batches of certificates with ACPC would take 1/480 of the bandwidth required by BCAM.

6.2 Processing costs

Another improvement of ACPC over BCAM is that, in the latter, the CAM is responsible for encrypting the batches received from the RA, after they are delivered by the PCA. Consequently, the CAM ends up encrypting certificates that have already been encrypted by the PCA: the latter process is intended to preserve the vehicles' privacy, and the former is executed for the purpose of preventing their activation in case of revocation. The proposed approach integrates both purposes into a single encryption process, performed by the PCA, which was already present in SCMS itself.

In terms of processing at the CAM, BCAM is such that the number of encryptions performed grows proportionally to the size of the batches, which depends on the size of pkg and on the number of certificates per time period (σ). In comparison, ACPC's design is such that each activation period requires one PRF execution combined with an elliptic curve scalar multiplication, for computing the corresponding blinded activation codes $A_t = f_a(\texttt{code}_{t,\mathrm{VID}}, t, \mathrm{VID}) \cdot G$. Therefore, many symmetric encryption operations can be replaced by a single computation of a blinded activation code. Since scalar multiplications over elliptic curves are known to be considerably more expensive than symmetric encryption, however, the actual benefits of ACPC depends on the system parameters. More precisely, for a given value of $|pkg|$, the larger the number of time periods covered by each activation period (α), the larger the number of encryptions replaced by the computation of a blinded activation code and, hence, the better the processing times of ACPC when compared to BCAM.

To give some concrete numbers, we have implemented our proposal using the RELIC cryptography library version 0.4.1 [2] running on an Intel i5 4570 processor. In this setting, the computation of one blinded activation code takes 242,822 cycles. This is approximately 300 times the cost of one single AES encryption, which takes only 832 cycles. Based on these numbers, Figure 7 shows the total processing costs at the CAM when we assume pseudonym certificate batches that span a period of 3-years, which corresponds to 156 one-week long time periods. The smallest number of encryption for BCAM, 12480, refers to a scenario where $\sigma = 20$ and $|pkg| = 512$ bits, meaning that $156 * 20 = 3120$ pseudonym certificates are encrypted by the CAM and each encryption takes $512/128 = 4$ calls to a 128-bit block cipher such as AES. The largest number of encryptions, 199680, is obtained when $\sigma = 40$ and $|pkg| = 4096$ bits, so the encryption of each of the $156 * 20 = 6240$ certificates requires $4096/128 = 32$ calls to the underlying 128-bit block cipher. As observed in this figure, the performance of ACPC is comparable to BCAM's, and usually surpasses it, in particular for larger numbers of α. For instance, when each activation period spans 4 certificate time periods, only $156/4 = 39$ blinded activation codes are computed, so the corresponding ACPC-4 curve always remains below BCAM's curve.

Even more importantly, the fact that the CAM does not re-encrypt batches in ACPC means that the vehicles do not need to remove this additional encryption layer when processing the received pseudonym certificates. Instead, the only decryption operation performed at the vehicle's side is the one originally present in SCMS itself, which reverses the encryption done by the PCA. Therefore, the total overhead incurred by ACPC refers to the computation of one (unblinded) activation code $f_a(\texttt{code}_{t,\mathrm{VID}}, t, \mathrm{VID})$ per activation period, so it can be included in the computation of the decryption key $\hat{e}_{t,c}$. When compared

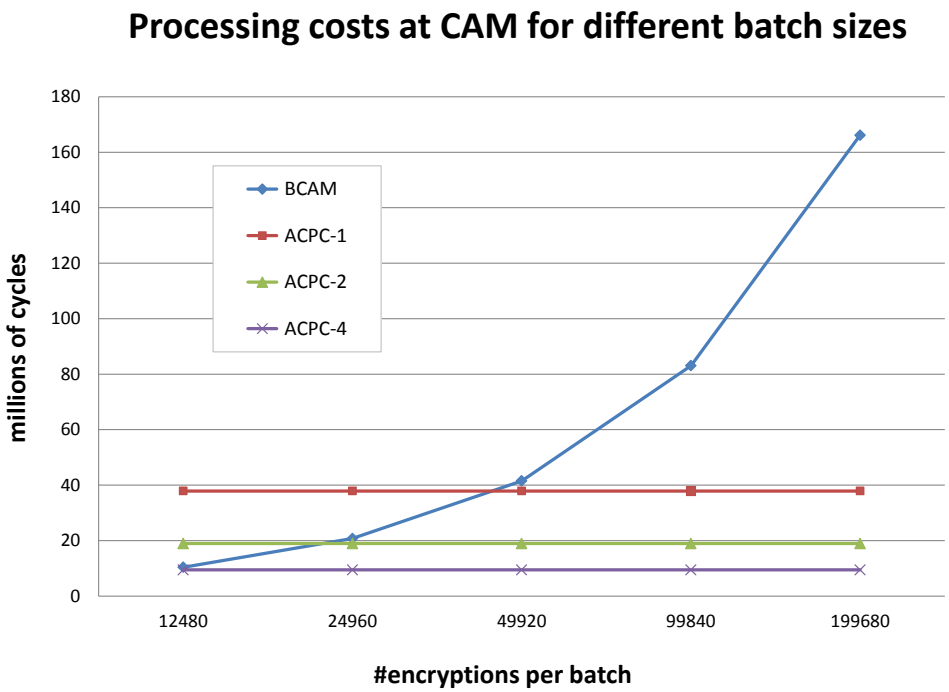

Figure 7: Processing costs at the Certificate Access Manager (CAM) for BCAM and ACPCẆe denote by ACPC-α the setting in which the number of time periods covered by each activation period is α.

to BCAM, this means that $|pkg|/128$ encryptions are replaced by τ/α invocations of a PRF at the vehicle's side. Even if we consider the scenario from Figure 7 in which the benefits of ACPC are less pronounced at the CAM's side (namely, $\alpha = 1$ and $|pkg| = 512$), this still means that ACPC trades 12480 decryptions for 156 PRF executions. In practice, assuming that one PRF execution takes roughly as much processing as one decryption, this means that the overhead added by ACPC's activation codes is approximately 1/80 of the overhead incurred by BCAM at the vehicles' side.

Finally, the usage of activation codes is transparent from the PCA's perspective, so its processing costs remain unchanged. For the RA, in turn, a single elliptic curve addition per activation period is necessary, to compute $E + A_t$, from which the corresponding public cocoon keys $\hat{E}_{t,c}$ can be obtained. Therefore, the overhead incurred by ACPC at the RA is expected to be negligible.

7 Conclusions

Security and privacy mechanisms are essential for preventing drivers from abusing V2X communications to gain unwarranted advantages over their peers, and also for limiting any entity's ability to track honest vehicles. A promising solution to address such requirements is the Security Credential Management System (SCMS), which provides efficient and privacy-preserving mechanisms for issuing pseudonym certificates to vehicles and revoking them in case of misbehavior. As a drawback, however, SCMS's revocation procedure is such that, after certificate identifiers are included in a CRL, it may take a long time for the corresponding CRL entries to be removed. This incurs not only in bandwidth overheads for the CRL distribution, but also increases the processing cost at vehicles for verifying a certificate's revocation status.

Aiming to address these issues, in this article we present improvements on SCMS's pseudonym certificate revocation process. The proposed design, named Activation Codes for Pseudonym Certificates (ACPC), is based on activation codes, small pieces of information without which pseudonym certificates previously issued become useless. Consequently, by ensuring that only non-revoked vehicles are periodically provided with those codes, entries associated with revoked vehicle's certificates can be safely removed from CRLs.

Even though ACPC builds upon ideas originally discussed in recent works, in particular IFAL [33] and BCAM [18], it leads to a significantly more security and efficient pseudonym certificate system. More precisely, SCMS's design is such that a dishonest PCA would have to collude with the RA to be able to track vehicles, and ACPC does not increase this vulnerability surface. In contrast, in IFAL a dishonest PCA could track vehicles without colluding with any other entities, whereas in BCAM a collusion between CAM and PCA would enable them to similarly violate the drivers' privacy, In terms of performance, one advantage of ACPC over IFAL refers to the distribution of activation codes: the CAM can broadcast codes to all non-revoked vehicles, so no bidirectional connectivity is required between vehicles and CAM. Even though such distribution process is as efficient as BCAM's, ACPC leads to a much more efficient certificate issuance process than BCAM: since the proposed design avoids the re-encryption of pseudonym certificates by the CAM, the CAM-RA bandwidth usage in ACPC is expected to be lower than 1/480 of BCAM's, and the processing overhead added by ACPCś activation codes at the vehicle's side is expected to be 1/80 or less than what is observed with BCAM.

Acknowledgments: This work was supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under grant 301198/2017-9, and by LG Electronics via the Foundation for the Technological Development of the Engineering Sciences (FDTE).

References

[1] W. Aiello, S. Lodha, and R. Ostrovsky. Fast digital identity revocation (extended abstract). In *Proc. of the 18th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'98)*, pages 137–152, London, UK, UK, 1998. Springer-Verlag.

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

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

[4] E. Biham. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. *Inf. Process. Lett.*, 84(3):117–124, nov 2002.

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

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

[7] G. Dimitrakopoulos and P. Demestichas. Intelligent transportation systems. *IEEE Vehicular Technology Magazine*, 5(1):77–84, March 2010.

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

[9] L. Figueiredo, I. Jesus, J. Machado, J. Ferreira, and J. Carvalho. Towards the development of intelligent transportation systems. In *Proc. of the IEEE Intelligent Transportation Systems (ITSC'2001)*, pages 1206–1211, 2001.

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

[11] J. Haas, Y. Hu, and K. Laberteaux. Design and analysis of a lightweight certificate revocation mechanism for vanet. In *Proc. of the 6th ACM International Workshop on VehiculAr InterNETworking*, pages 89–98. ACM, 2009.

[12] J. Haas, Y. Hu, and K. Laberteaux. Efficient certificate revocation list organization and distribution. *IEEE Journal on Selected Areas in Communications*, 29(3):595–604, March 2011.

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

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

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

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

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

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

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

[20] 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.

[21] D. McGrew, M. Curcio, and S. Fluhrer. Hash-based signatures. Internet-Draft draft-mcgrew-hash-sigs-06, Internet Engineering Task Force, mar 2017. Work in Progress.

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

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

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

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

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

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

[28] B. Preneel. *Encyclopedia of Cryptography and Security: Davies–Meyer Hash Function*, pages 136–136. Springer US, Boston, MA, 2005.

[29] M. Raya, P. Papadimitratos, I. Aad, D. Jungels, and J-P. Hubaux. Eviction of misbehaving and faulty nodes in vehicular networks. *IEEE Journal on Selected Areas in Communications*, 25(8), 2007.

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

[31] M. Simplicio, E. Cominetti, H. Patil, J. Ricardini, L. Ferraz, and M. Silva. A privacy-preserving method for temporarily linking/revoking pseudonym certificates in vehicular networks. Cryptology ePrint Archive, Report 2018/185, 2018. https://eprint.iacr.org/2018/185.

[32] M. Simplicio, E. Cominetti, H. Kupwade Patil, J. Ricardini, and M. Silva. The unified butterfly effect: Efficient security credential management system for vehicular communications. IACR eprint archive: https://eprint.iacr.org/2018/089.pdf, 2018.

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

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

A Birthday attack against BCAM's hash trees

The structure of BCAM's binary hash trees is such that their k-bit nodes are computed via iterative hashing, using a constant suffix for each branch. More precisely, starting from a random root $\texttt{node}^t(0,0)$, each node $\texttt{node}^t(depth, count)$ of $\texttt{tree}^t$ is computed from its parent as follows:

$$\texttt{node}^t(depth, count) = Hash(\texttt{node}^t(depth-1, \lfloor count/2 \rfloor) \,\|\, b^p),$$

where $b = 0$ (resp. $b = 1$) if the node is a left (resp. right) child, and $p \geqslant 1$ is a suitable padding length. For example, when $k = 256$ and the hash function employed is SHA-256,

adopting $1 \leqslant p < 192$ would allow the underlying compression function to be called only once when computing any node of the tree.

Suppose that a vehicle with identifier VID_r is revoked. In that case, the leaf $\texttt{node}^t(|\text{VID}|, VID_r)$ should not be computed from the message broadcast by the CAM, for every future value of t. This means that the set N_r of all nodes in the path between the root and that leaf must remain secret. To accomplish this, the CAM only broadcasts *children* of the nodes in N_r. For example, as mentioned in Section 4.3, the revocation of $\texttt{node}^t(3,4)$ leads to the disclosure of the set $N_d = \{\texttt{node}^t(1,0), \texttt{node}^t(2,3), \texttt{node}^t(3,5)\}$. As long as the tree is built using a secure hash function, it is not straightforward to use any node in N_d to compute nodes in the set $N_r = \{\texttt{node}^t(0,0), \texttt{node}^t(1,1), \texttt{node}^t(2,2), \texttt{node}^t(3,4)\}$. Indeed, doing so corresponds to finding pre-images for nodes in the set N_d.

To overcome the security of BCAM's activation trees, the following attack strategy can be employed to recover activation codes for revoked vehicles. First, the attacker picks an arbitrary, k-bit long $\texttt{link}^0$, and arbitrarily chooses between $b = 0$ or $b = 1$. The value of $\texttt{link}^0$ is then used as the anchor for a hash chain of the form $\texttt{link}^j = \mathit{Hash}(\texttt{link}^{j-1} \parallel b^p)$, until 2^n hashes are performed. For simplicity, we assume that no collision occurs during this process, i.e., that $\texttt{link}^j \neq \texttt{link}^{j'}$ for all $j \neq j'$. Nevertheless, this simplification comes without loss of generality because, whenever there is a collision, the attacker could simply (1) save the current chain, (2) pick a new anchor distinct from any previously computed $\texttt{link}^j$, and then (3) start a new chain from this anchor. Actually, picking different anchors for building multiple chains is likely advantageous anyway, because this facilitates the parallel processing of hashes. As long as 2^n different hashes are made available in this manner, the attack can proceed.

Due to the birthday paradox, an attacker that gathers 2^m nodes disclosed by the CAM has a high probability to find a match between at least one of those nodes and some of the 2^n previously computed $\texttt{link}^j$ if $m + n \geqslant k$. Suppose that a match occurs between $\texttt{link}^j$ and $\texttt{node}^t(\mathit{depth}, \mathit{count})$. In this case, $\texttt{link}^{j-1}$ is a valid pre-image for $\texttt{node}^t(\mathit{depth}, \mathit{count})$ with padding b^p. Hence, if the attacker picked $b = 0$ and $\texttt{node}^t(\mathit{depth}, \mathit{count})$ is a left child, it is very likely that $\texttt{link}^{j-1}$ will match the parent of $\texttt{node}^t(\mathit{depth}, \mathit{count})$ in the activation tree — unless $\texttt{link}^{j-1}$ is a second pre-image rather than the actual pre-image. If the parent of $\texttt{node}^t(\mathit{depth}, \mathit{count})$ is also a left child, its own parent is also likely to match $\texttt{link}^{j-2}$, and so forth. An analogous argument applies if $b = 1$ and $\texttt{node}^t(\mathit{depth}, \mathit{count})$ is a right child. As a result, such collisions have roughly 50% of chance of giving the attacker access to nodes belonging to the revoked set N_r. All certificates whose revocation depended on those nodes can then be activated.

Considering this attack scenario, the growth of the number of revoked devices has two negative effects on the system's security. First, the recovery of one node from the set N_r becomes more likely to give access to activation codes of multiple revoked devices. The reason is that a node in a given position of the tree always allow the computation of a same number of leafs (the lower the depth, the higher this number). When the number of revoked devices increase, so does the number of leaves covered by that node that should remain concealed to prevent the corresponding activation codes from being recovered. Second, the number of nodes disclosed by the CAM that would lead to useful collisions also grows, i.e., the value of m becomes larger.

Since such attacks trade time for space, one possible defense strategy is to adopt a large enough k parameter. For example, the authors of BCAM suggest $k = 256$ (cf. [18], Section 4.1.3), meaning that the attacker would have to compute, say, $2^n = 2^{128}$ hashes and then gather $2^m = 2^{128}$ nodes from the CAM before a collision actually occurs. Therefore, in practice, the attacks hereby described do not pose an actual security threat to BCAM.

Nevertheless, there is a more efficient defense strategy for this issue, originally discussed by Leighton and Micali [20] in the context of hash-based signatures [21] and also proposed for use with SCMS's linkage trees [31]: to use a different suffix for each node computation. This

strategy comes from the observation that collisions between $\texttt{link}^{\texttt{j}}$ and $\texttt{node}^t(depth, count)$ are useless if they are computed with different suffixes. After all, in that case $\texttt{link}^{\texttt{j}-1}$ will not match the parent of $\texttt{node}^t(depth, count)$, i.e., it will necessarily be a second pre-image rather than the actual pre-image of that node. At the same time, attackers are unable to gather more than 1 value of $\texttt{node}^t(depth, count)$ for any given suffix. Consequently, to obtain a high probability of collisions for that suffix, the attacker would have to build a table with $2^n = 2^{k-m} = 2^k$ entries. In other words, the approach adopted in ACPC leads to a 128-bit security level even when the nodes themselves are 128-bit long.