A Security Credential Management System for V2V Communications

William Whyte*, André Weimerskirch†, Virendra Kumar*, Thorsten Hehn‡
*{wwhyte, vkumar}@securityinnovation.com
†andre.weimerskirch@escrypt.com
‡thorsten.hehn@vw.com

***Abstract*—We present a security credential management system for vehicle-to-vehicle communications, which has been developed under a Cooperative Agreement with the US Department of Transportation. This system is currently being finalized, and it is the leading candidate design for the V2V security backend design in the US, subject to review by the US Department of Transportation and other stakeholders. It issues digital certificates to participating vehicles for establishing trust among them, which is necessary for safety applications based on vehicle-to-vehicle communications. It supports four main use cases, namely, bootstrapping, certificate provisioning, misbehavior reporting and revocation. The main design goal is to provide both security and privacy to the largest extent reasonable and possible. To achieve the latter, vehicles are issued pseudonym certificates, and the provisioning of those certificates is divided among multiple organizations. One of the main challenges is to facilitate efficient revocation while providing privacy against attacks from insiders.**

I. Introduction

Vehicle-to-Vehicle (V2V) communications among nearby vehicles in the form of continuous broadcast of Basic Safety Messages (BSMs) has the potential to prevent up to 75% of all roadway crashes through active safety applications [1]. Following a series of field operational tests, the US Department of Transportation (USDOT) is expected to announce in 2013 whether it plans to seek a mandate for V2V communications equipment, to be included in all new build vehicles after a particular date.

The correctness and reliability of BSMs, which contain information like sender's position, speed, etc., are of prime importance as they directly affect the outcome and effectiveness of safety applications based on them. To prevent an attacker from inserting false messages, most studies recommend that the sending vehicles digitally sign each BSM, and the receiving vehicles verify the signature before acting on it [2], [3], [4], [5], [6], [7], [8].

A Public-Key Infrastructure (PKI) that facilitates and manages digital certificates is necessary for building trust among participants and for proper functioning of the system. Our Security Credential Management System (SCMS) implements a PKI with some additional new features. This SCMS is currently the leading candidate design for the V2V security backend design in the US, subject to review by the USDOT and other stakeholders. It is distinguished from a traditional PKI in several aspects, the two most important ones being its size (i.e., the number of vehicles that it supports) and the balance among security, privacy, and efficiency. At its full capacity, assuming 300 million vehicles, it will issue approximately 300 billion certificates per year[1]. The largest current PKI, deployed by the US Department of Defense, is several orders of magnitude smaller and issues under 10 million certificates per year. At the core of its design, the proposed SCMS has several novel cryptographic constructs to provide a high level of security and privacy to the users while keeping the system very efficient. As a result, the presented SCMS design is significantly different from any previously implemented PKI due to the underlying security objectives and size, however, it is somewhat similar to the design of the European V2X PKI [2]. The main differences to [2] include an increased focus on privacy against attacks from SCMS insiders, efficient handling of revocation, and an efficient method for updating certificates based on the butterfly key expansion algorithm. An early version of the proposed SCMS has already been implemented, operated and tested [9].

II. SCMS Design Overview

In this section, we present the design of the proposed SCMS by briefly explaining its components and then discussing the rationale behind the design. We say that an SCMS component is *intrinsically-central*, if it can have exactly one distinct instance for proper functioning. A component is *central*, if it chosen to have exactly one distinct instance in the considered instantiation of the system. Distinct instances of a component have different identifiers and do not share cryptographic materials. Central components do support load balancing. Figure 1 gives an overview of the overall system architecture. The lines connecting different SCMS components in that figure are relationship lines, meaning that one component sends information or certificates to the other. Additionally, there is a bold line connecting a device to the SCMS, to indicate out-of-band secure communication. The conceptual model of the SCMS is the most flexible and full-featured system. It can be simplified at the expense of losing some flexibility, e.g. making all the components *central*. We considered two deployment models for initial and full deployment stages, respectively. Due to space limitations, we focus only on the full deployment model here.

A. Threat Models and Application Concepts

Besides its standard functionality as a PKI system, the SCMS is designed to handle the following types of attacks.

- Attacks on end-users' privacy from SCMS outsiders

[1]This number may be even greater if pedestrian and cyclist-borne units become part of the system.

- Attacks on end-users' privacy from SCMS insiders
- Authenticated messages leading to false warnings

We address the first two items by "Privacy by Design". The third item is addressed by revocation, which uses misbehavior detection and reporting scheme to identify devices to revoke.

1) Privacy by Design: A key goal of the system is to protect the privacy of end-users. Since most vehicles are primarily operated by a single user, the system should be designed to make tracking hard, i.e. it should difficult for two eavesdroppers in physically distant locations to tell whether two transmissions came from the same vehicle. To maintain privacy against attackers from outside the SCMS, we propose frequent certificate changes (e.g. every 5 minutes). Another key requirement is that these attacks should be difficult to mount for SCMS insiders. In order to do so, the SCMS operations are divided among its different components, and those components are required to have organizational separation between them. At a high level, the SCMS operations are divided such that at least two SCMS components need to collude to gain enough information for tracking a device. A device sending BSMs may potentially be tracked in other ways that are not addressed by the SCMS: by its RF fingerprint [10], by deploying a large-scale sniffer network, etc.

2) Misbehavior Detection & Revocation: The SCMS supports revocation by distributing a Certificate Revocation List (CRL), which are used to reject certificates from a misbehaving device. In addition, the SCMS maintains a *blacklist*, which is internal to the SCMS, to deny future certificate requests by revoked devices. Misbehavior detection is used to identify devices that need to be revoked. The novel concept of *linkage values* (cf. Section IV-B) allows for efficient revocation.

B. Full SCMS Structure

Below we briefly describe the SCMS components[2] as shown in Figure 1. The figure shows different components within the system by their logical roles. An implementation of the system may combine multiple logical roles within a single organization with proper separation of the logical roles.

- SCMS Manager: Ensures efficient and fair operation of the SCMS, sets guidelines for reviewing misbehavior and revocation requests to ensure that they are correct according to procedures.
- Certification Services: Provides information on which types of devices are certified to receive digital certificates and specifies the certification process.
- CRL Store (CRLS): Stores and distributes CRLs. This is a simple pass-through function since CRLs are signed by the CRL Generator.
- CRL Broadcast (CRLB): Broadcasts the current CRL, may be done through Road Side Equipment (RSEs) or satellite radio system, etc. This is a pass-through function.
- Device: An end-entity device that sends BSMs, for example On-Board Equipment (OBE) or After-market Safety Device (ASD).
- Device Configuration Manager (DCM): Provides authenticated information about SCMS component configuration changes to devices, which may include a component changing its network address or certificate, or relaying policy decisions issued by the SCMS Manager. It is also used to attest to the Enrollment CA that a device is eligible to receive enrollment certificates.
- Enrollment CA (ECA): Issues enrollment certificates, which act as a passport for the device and can be used to request pseudonym certificates. Different ECAs may issue enrollment certificates for different geographic regions, manufacturers, or device types.
- Linkage Authority (LA): Generates linkage values, which are used in the certificates and support efficient revocation. There are two LAs in the SCMS, referred to as LA_1 and LA_2. The splitting prevents the operator of an LA from linking certificates belonging to a particular device.
- Location Obscurer Proxy (LOP): Hides the location of the requesting device by changing source addresses, and thus prevents linking of network addresses to locations. Additionally, when forwarding information to the Misbehavior Authority (MA), the LOP shuffles the reports to prevent the MA from determining the reporters' routes.
- Misbehavior Authority (MA): Processes misbehavior reports to identify potential misbehavior by devices, and if necessary revokes and adds devices to the CRL. It also initiates the process of linking a certificate identifier to the corresponding enrollment certificates, and adding the enrollment certificate to an internal blacklist. The MA contains three subcomponents: Internal Blacklist Manager (IBLM), which sends information required for updating the internal blacklist to the RA; Global Detection (GD), which determines which devices are misbehaving; and CRL Generator (CRLG), which issues certificate revocation lists to the outside world.
- Pseudonym CA (PCA): Issues short-term (pseudonym) certificates to devices. Individual PCAs may, for example, be limited to a particular geographic region, a particular manufacturer, or a type of devices.
- Registration Authority (RA): Validates, processes, and forwards requests for pseudonym certificates to PCA.
- Request Coordination (RC): Ensures that a device does not request more than one set of certificates for a given time period. It coordinates activities between different RAs, and is only needed if a device could request certificates from multiple RAs.

C. Pseudonym Certificate Provisioning Model

We need a pseudonym certificate provisioning model that provides an appropriate balance among several conflicting requirements:

- **Privacy vs. Size vs. Connectivity**: Certificates should be used only for short periods of time for privacy, but the devices can neither store a large number of certificates, nor do they have frequent connectivity to the SCMS to download certificates on demand.

[2]For space considerations, we skip the standard components of a PKI, such as Root CA, Intermediate CA, etc.

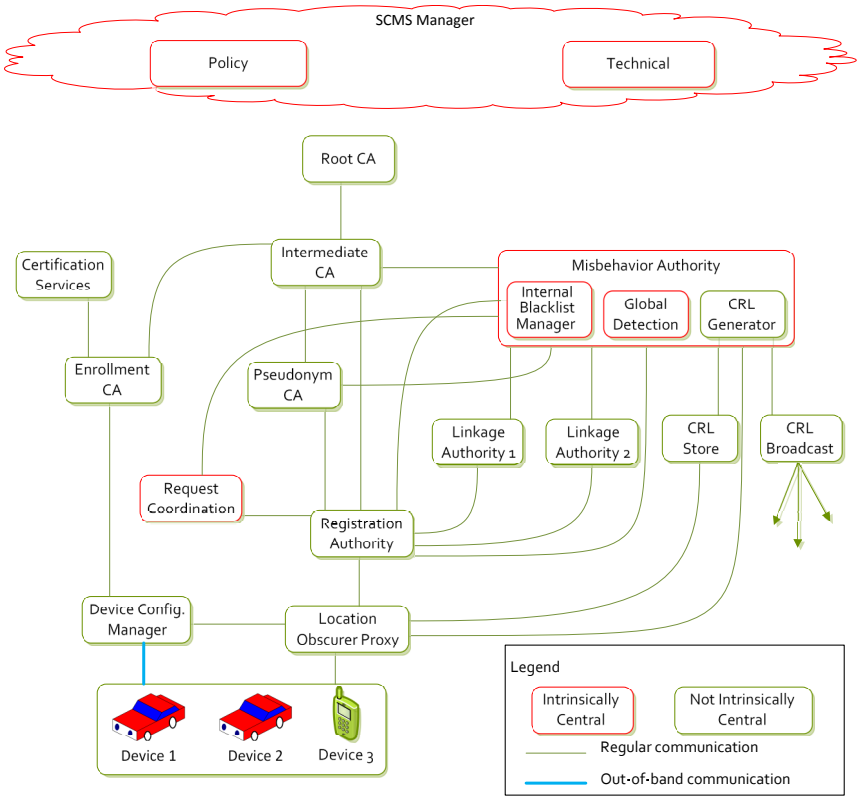

Fig. 1. Overall System Architecture

- **CRL Size vs. Retrospective Unlinkability**: The SCMS should be able to revoke misbehaving devices[3], but putting all valid certificates of a device on the CRL would make it very large. We need a mechanism to revoke a large number of certificates efficiently, but this must not reveal certificates that were used by a device before it started misbehaving.
- **Certificate Waste vs. Sybil Attack**: Certificates must be changed periodically for privacy. One option is to have a large number of certificates, each valid one after the other for a short period of time. This would result in a large number of unused certificates[4]. Another option is to have multiple certificates valid simultaneously for longer time periods. This would enable masquerading as multiple devices by compromising a single device (the so-called *Sybil attack* [11]).

In the Safety Pilot [9] model, a certificate was valid for a specific 5-minute period, and the devices were given 3 years' worth of certificates, amounting to more than $300,000$ certificates. This approach is prohibitively expensive in terms of automotive-grade storage requirements on the device. We studied different variants of this model, but none of them offer a good balance among all the properties listed above. Instead, we decided to adopt the model used by CAR 2 CAR Communication Consortium (C2C-CC) [2], with certain modifications to suit our requirements.

In the C2C-CC model, multiple certificates are valid in a given time period, the certificate validity period is days rather than minutes, and the certificate usage pattern can vary from device to device, e.g. a device could use a certificate for 5 minutes after start-up, then switch to another certificate, and use that either for 5 minutes, or until the end of the journey. This model offers enough flexibility to find a good balance among our different requirements except "CRL size vs. retrospective unlinkability", which we address by our novel construct of linkage values (cf. Section IV-B). In particular, our proposal is to use the C2C-CC model with the following parameter values:

- Certificate validity time period: 1 week
- Certificates valid simultaneously (batch size): $20 - 40$
- Overall covered time-span (super-batch size): $1 - 3$ years

These parameters can be further fixed by the SCMS Manager so that devices are compatible with each other. Two points are worth noting:

1) This model provides a reasonable level of privacy against tracking while keeping the storage requirements low due to a high utilization of certificates (e.g., far fewer certificates are wasted compared to the Safety Pilot

[3]CRLs can be avoided, if devices were required to request new certificates frequently such that misbehaving devices could be refused new certificates. This would require frequent connectivity to the SCMS, which may not be a reasonable assumption.

[4]Based on the US Census Bureau's annual American Community Survey for 2011, the average daily commute time is less than 1 hour, so more than 95% of certificates would be wasted.

Model Deployment). A device that uses fewer than the number of certificates granted for a week cannot be linked. Moreover, if a device re-uses certificates, it is linkable only within a week. Privacy-conscious users could potentially buy additional certificates so long as their device had the storage space and demonstrated that it had appropriate physical security against compromise.

2) It allows for an easy topping-off mechanism (without losing any other benefits) of certificates at a granularity level of certificate validity period within the life cycle of a super-batch. For example, if a device comes to the dealer one year after the last certificate loading, it is easy to delete the outdated certificates, freeing up space to provide it with another year's worth of certificates.

III. Butterfly Key Expansion

Butterfly keys are a novel cryptographic construction that allow a device to request an arbitrary number of certificates, each with different signing keys and each encrypted with a different encryption key, using a request that contains only one verification public key seed, one encryption public key seed, and two expansion functions. Without butterfly keys, the device would have to send a signing key and a unique encryption key for each certificate. Butterfly keys reduce upload size, allowing requests to be made when there is only suboptimal connectivity, and also reduce the work to be done by the requester to calculate the keys. Butterfly key expansion is described below for elliptic curve cryptography, but it could easily be adapted to any discrete log-based problem. In the following, we denote integers by lower-case characters and curve points by upper-case characters. The elliptic curve discrete logarithm problem is basically the statement: Given P and $A = aP$, but not a, it is hard to compute the value of a [12].

Butterfly keys make use of this as follows. There is an agreed base point, called G, of some order l. The *caterpillar keypair* is an integer, a, and a point $A = aG$. The certificate requester provides the RA with A and with an expansion function, $\mathrm{f}_k(\iota)$, which is a pseudo-random permutation in the integers mod l. For example, for points on the NIST curve NISTp256 [13], $\mathrm{f}_k(\iota)$ could be defined as $\mathrm{AES}_k(0^{128} \oplus \iota)||\mathrm{AES}_k(1^{128} \oplus \iota)$, where AES is the Advanced Encryption Standard block-cipher, $\iota < 2^{128}$, and x^y, $x \in \{0,1\}$, refers to an array of values x of length y; this means the expansion function is defined by k. Now RA can generate up to 2^{128} *cocoon public keys* as $B_\iota = A + \mathrm{f}_k(\iota) * G$, where the corresponding private keys will be $b_\iota = a + \mathrm{f}_k(\iota)$, so the public keys are known to the RA but the private keys are known only to the device. The RA includes the cocoon public keys in the certificate requests sent to the PCA.

If these expanded public keys were used unaltered by the PCA, the RA, which knows which public keys come from a single request, could recognize those public keys in the certificates and track the holder. To avoid this, for each input public key B_i, the PCA generates a random c and obtains $C = cG$. The *butterfly public key* which is included in the certificate is $B_\iota + C$. The PCA returns both the certificate and the private key reconstruction value c to the RA to be returned to the device[5]. To prevent the RA from working out which certificate corresponds to a given public key in a request, the certificate and the reconstruction value c must be encrypted. To prevent the PCA from knowing which certificates go to which vehicle, each certificate must be encrypted with a different key. The encryption keys are also generated with the butterfly key approach: the device provides a caterpillar public key $H = hG$, the RA expands it into cocoon public encryption keys $H + J_\iota = H + f_e(\iota)G$, and the PCA uses these keys to encrypt the response.

IV. Use Cases

The SCMS supports four main use cases: device bootstrap, pseudonym certificate provisioning, misbehavior reporting, and global misbehavior detection & revocation.

A. Device Bootstrap

Bootstrap is executed at the start of the life cycle of a device. It equips the device with all the information required to communicate with the SCMS and with devices. Bootstrap must provide correct information, and the CAs must issue certificates only to certified devices. Any bootstrap process is acceptable that results in this information being established securely. We anticipate that the DCM will establish a secure channel with the SCMS, and will communicate with the device to be bootstrapped in a secure environment. Bootstrap consists of two logical operations, initialization and enrollment. Initialization is the process by which the device obtains certificates it needs to trust received messages. Enrollment is the process by which the device obtains certificates it will need to send messages. Information received in the initialization process includes (1) the certificate of all root CAs and possibly of intermediate CAs as well as PCAs, (2) the certificate of the misbehavior authority and the CRL generator to report misbehavior and learn about revocation, and (3) the certificate of the DCM. In the enrollment process, information required to actively participate includes (1) the enrollment certificate, (2) the certificate of the ECA, and (3) the certificate of the RA and information necessary to locate the RA. During the enrollment process, the certification services provide the ECA with information about device models which are eligible for enrollment. This requires that the ECA receives trustworthy information about the type of the device to be enrolled, from the device itself or from the DCM.

B. Pseudonym Certificate Provisioning

The pseudonym certificate provisioning process is illustrated in Figure 2, and is designed to protect privacy from inside attackers. The SCMS is designed to ensure that no single component knows or creates a complete set of information

[5]This description covers explicit certificates, i.e. certificates that include the public key explicitly. In fact, the SCMS as currently implemented issues *implicit certificates* [14]. The implicit certificate generation process inherently changes the public key in a way that leaves input and output uncorrelatable by the RA and is consistent with the motivation for butterfly keys.

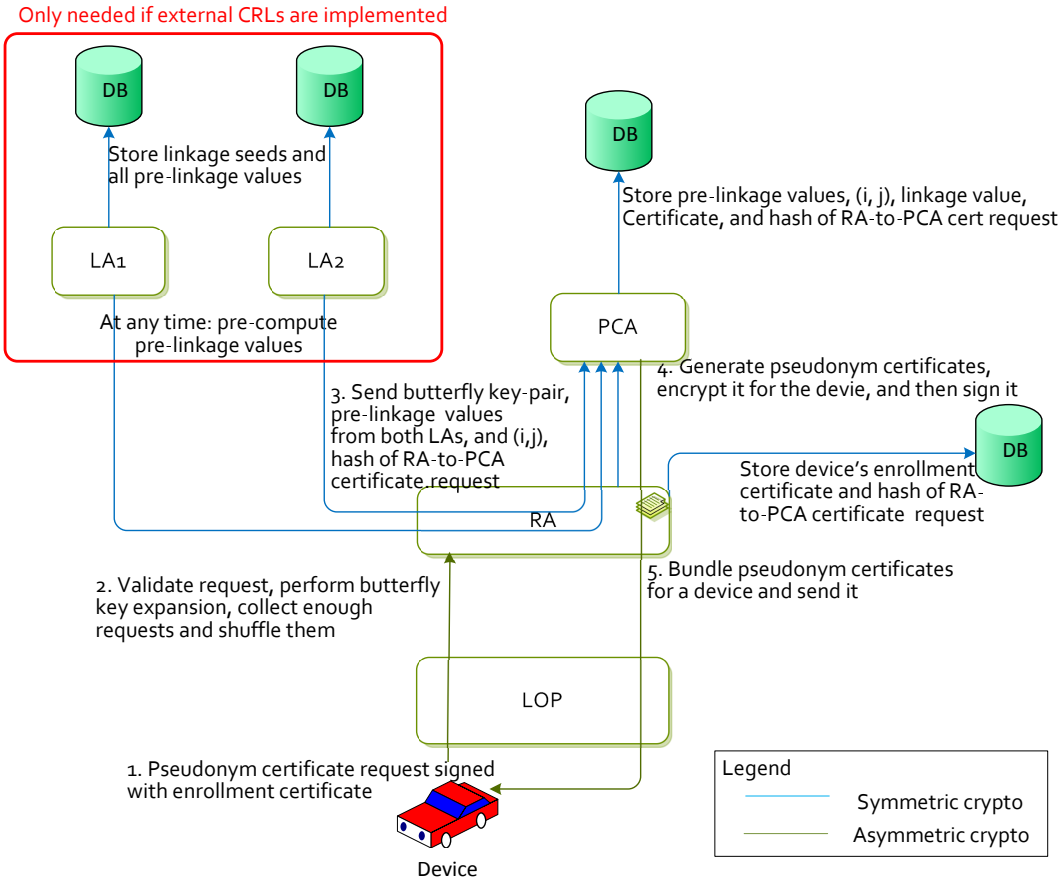

Fig. 2. Certificate Provisioning

that would enable tracking of a vehicle. For example, the RA knows the enrollment certificate of a device that requests pseudonym certificates, but even though the pseudonym certificates are delivered to the device by the RA, it never gets to see the content of those certificates; the PCA creates each individual pseudonym certificate, but it doesn't know the recipient of those certificates, nor does it know which certificates went to the same device. Privacy mechanisms in the SCMS include:

- **Obscuring Physical Location.** The LOP obscures the physical location of an end-entity device to hide it from the RA and the MA.
- **Hiding Certificates from RA.** The *butterfly key expansion* process ensures that the public keys in requests cannot be correlated with the public keys in the resulting certificates. More details are given in Section III.
- **Hiding Batch from PCA.** The RA splits incoming requests of devices into requests for single certificates, and shuffles requests of all devices before sending them to the PCA. This prevents the PCA from figuring out if two different certificates went to the same device, which would violate our privacy goal by enabling the PCA to link certificates. The RA may, for example, aggregate and shuffle all requests received in a six month period.
- **Linkage Values.** For any set of pseudonym certificates provided to a device, the SCMS inserts *linkage values* that can be used to revoke all of the certificates with validity equal to or later than some time i. These linkage values are computed by XORing the pre-linkage values generated by the Linkage Authorities LA_1 and LA_2, and can be generated in advance of the request for a set of pseudonym certificates. Let $\mathrm{la\_id}_1, \mathrm{la\_id}_2$ be 32-bit identity strings associated with LA_1, LA_2, respectively. For a set of certificates, first the LA_1 (resp., the LA_2) picks a random 128-bit string called the initial linkage seed $\mathrm{ls}_1(0)$ (resp., $\mathrm{ls}_2(0)$), then for each time period (e.g., a week) $i > 0$ calculates the linkage seed $\mathrm{ls}_1(i) = \mathrm{H}(\mathrm{la\_id}_1 \| \mathrm{ls}_1(i-1))$ (resp., $\mathrm{ls}_2(i) = \mathrm{H}(\mathrm{la\_id}_2 \| \mathrm{ls}_2(i-1))$). In this coherence, $\mathrm{H}(m)$ denotes the 16 most significant bytes of the SHA-256 hash output on m, and for bit-strings x and y, $x \| y$ denotes their bit-wise concatenation. Furthermore, we will use $\mathrm{E}(k, m)$ to denote the 8 most significant bytes of AES-128 block-cipher operation on message m with key k. For each value $j > 0$ (which specifies the number of overlapping certificates in a given time period, e.g., 20), the LA_1 (resp., the LA_2) calculates the pre-linkage value $\mathrm{plv}_1(i, j) = \mathrm{E}(\mathrm{ls}_1(i), \mathrm{la\_id}_1 \| j)$ (resp., $\mathrm{plv}_2(i, j) = \mathrm{E}(\mathrm{ls}_2(i), \mathrm{la\_id}_2 \| j)$). Pre-linkage values are

encrypted (individually) for the PCA but sent to the RA for association with a certificate request.

- **Hiding Linkage Information.** The PCA computes the linkage value to be included in a certificate by XORing together the two pre-linkage values from the LAs, which are generated independently by the two LAs and encrypted for the PCA to prevent the RA from colluding with one of the LAs and mapping pre-linkage values to linkage values. Therefore, no single component is able to link pseudonym certificates of a single device.

Below we present a detailed step-by-step description of the pseudonym certificate provisioning process.

- **Step 1.** When LOP receives a request (signed with the device's enrollment certificate, containing a public Butterfly Key seed, and encrypted to the RA) from a device for a specified time period, it obscures the device's identifiers (e.g., IP address), and forwards it to the RA.
- **Step 2.** The RA first decrypts the request, and checks if the signature and the device's enrollment certificate are valid and that the latter is not revoked, and then also checks (with the help of RC, if necessary) if this is the only request by the device for that particular time period. If all checks succeed, the RA sends an acknowledgement to the device, and performs the butterfly key expansion as explained in Section III. Otherwise, the request is denied. The RA collects several such requests from different devices along with the sets of pre-linkage values received from the LAs. Once enough such requests are available, the RA shuffles them. Note that in follow-up requests by a device, the RA might request pre-linkage values from each of the LAs for a particular initial linkage seed that is associated with that device.
- **Step 3.** The RA sends requests for pseudonym certificates to the PCA, for one certificate per request, where each request consists of a butterfly key-pair, an encrypted pre-linkage value from each of the LAs $(\mathrm{plv}_1(i,j), \mathrm{plv}_2(i,j))$, the certificate validity time period i and index j, and the hash of the request.
- **Step 4.** The PCA completes the butterfly key expansion hiding the certificate's public key from the RA. It then generates a pseudonym certificate with linkage value $\mathrm{lv}(i,j) = \mathrm{plv}_1(i,j) \oplus \mathrm{plv}_2(i,j)$, encrypts the private key reconstruction value and the certificate for the device, signs the encrypted packet, and sends it to the RA[6].
- **Step 5.** The RA collects and bundles the encrypted pseudonym certificates and the corresponding private key reconstruction values for a given device and delivers it via LOP to the device. These bundles are called super-batch.

For revocation purposes, the SCMS components involved in the pseudonym certificate provisioning store different information corresponding to a given pseudonym certificate as listed in Table I.

[6]Signing the encrypted packet provides a guarantee to the device that the PCA encrypted the packet with the device's public key. This prevents a man-in-the-middle attack where an insider at the RA substitutes their public key for the valid response encryption key, which is also known to the RA.

TABLE I
INFORMATION STORED BY SCMS COMPONENTS

Component	Information Stored
LA	Initial linkage seed, pre-linkage value
PCA	Pre-linkage values from both LAs and their corresponding (i,j) values, linkage value, certificate, and hash of RA-to-PCA pseudonym certificate request
RA	Enrollment certificate and its validity period, hash value of RA-to-PCA pseudonym certificate request

C. Misbehavior Reporting

Devices will send misbehavior reports to the MA via the LOP, which will, additionally to obscuring the source, shuffle the reports from multiple reporters to prevent the MA from reconstructing the reporter's path based on the reports. The format of a misbehavior report is not fully defined yet, but a report will potentially include reported (suspicious and alert-related) BSMs as well as the reporter's signature and certificate, and will be encrypted by the reporter for the MA. Note: A device will also have misbehavior detection algorithms which work on a local level.

D. Global Misbehavior Detection & Revocation

The algorithms of global misbehavior detection have not been developed yet, but the interface to the other components, which allows for obtaining linkage information, is already specified. Linkage information is required at the MA to find whether multiple misbehavior reports point to the same device. The following actions are required from the components:

1) The PCA and both the LAs have to collaborate to determine external revocation information for the CRL.
2) The PCA and the RA have to collaborate to determine the enrollment certificate of the misbehaving device for the internal blacklist.

Below we present a detailed step-by-step description of the process of identifying the linkage seeds and the enrollment certificate corresponding to a pseudonym certificate as illustrated in Figure 3. Some of the communications in the steps below need to be digitally signed.

- **Step 1.** The MA receives misbehavior reports, including a reported pseudonym certificate with linkage value $\mathrm{lv} = \mathrm{plv}_1 \oplus \mathrm{plv}_2$.
- **Step 2.** The MA runs global detection algorithms to determine which reported pseudonym certificates are of interest, i.e. whose linkage seeds and the corresponding enrollment certificates need to be determined.
- **Step 3.** The MA makes a request (signed) to the PCA to map the linkage value of the identified pseudonym certificate, lv, to the corresponding pre-linkage values $(\mathrm{plv}_1, \mathrm{plv}_2)$ and the hash value of the RA-to-PCA pseudonym certificate request, all from the PCA's database. The PCA returns these values to the MA.
- **Step 4.a.** The IBLM sends the hash value of the RA-to-PCA pseudonym certificate request (signed) to the RA so that it can add the corresponding enrollment certificate

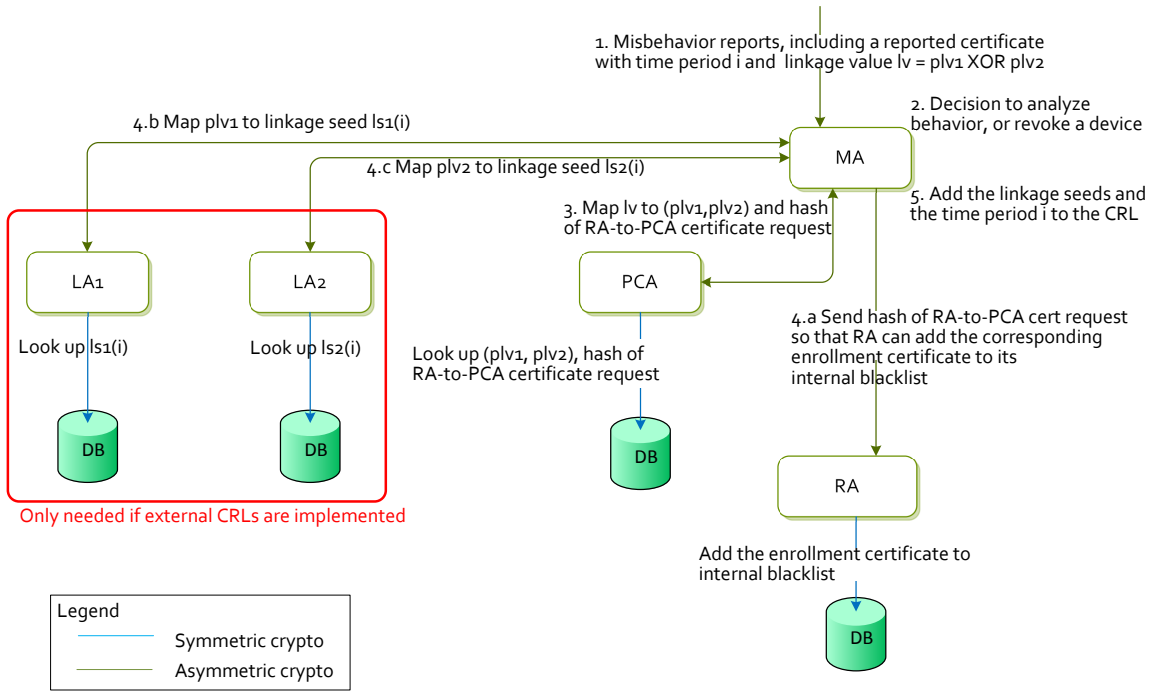

Fig. 3. Revocation

to the internal blacklist. The RA does not return a value, i.e., does not give the enrollment certificate to anyone. If a device is allowed to make pseudonym certificate requests to multiple RAs, the RC needs to manage the internal blacklist. For simplicity, the RC is not shown in Figure 3.

- **Steps 4.b., 4.c.** The MA makes a request to the LA_1 (resp., the LA_2) to map plv_1 (resp., plv_2) to the linkage seed $\text{ls}_1(i)$ (resp., $\text{ls}_2(i)$), where i is the currently valid time period. Both the LAs return the linkage seed to the MA. Note that given a linkage seed $\text{ls}_1(i)$, only the *forward* linkage seeds (i.e., $\text{ls}_1(j)$ for $j \geq i$) can be calculated, and thus *backward* privacy of the revoked device is maintained.
- **Step 5.** The linkage seeds $\text{ls}_1(i)$ and $\text{ls}_2(i)$, and the time period i are added to the CRL. When the next CRL is due, the CRLG signs the CRL and publishes it.

The size of the CRL grows linearly with the number of entries. We think that the size of the CRL needs to be upper-bounded, and that the entries need to be ordered in terms of priority. Note that currently there is no way to undo a revocation, and a revoked device can be reinstated only by repeating the process of bootstrapping, cf. Section IV-A.

V. Organizational Separation

Different SCMS components represent different logical functions. For privacy, some distinct logical functions must be provided by distinct organizations. This section identifies which SCMS components must be organizationally separate. The general rule is that two components cannot be run by the same organization if the combined information held by the components would allow an insider to track a vehicle.

- **PCA and RA:** If these two components were run by one organization, the organization would know which pseudonym certificates had been issued to which device. This is because the RA knows the requests to which certificates correspond, and the PCA knows the corresponding pseudonym certificates.
- **PCA and one of the LAs:** If an organization ran the PCA and either (or, both) of the LAs, it could link all pseudonym certificates (from a super-batch) issued to any device since LA knows a set of pre-linkage values that go into the certificate set, and PCA sees these pre-linkage values at certificate generation time.
- **LA_1 and LA_2:** If an organization ran both the LAs, it would know all the pre-linkage values and XOR them opportunistically to obtain the linkage values, which appear in plaintext in the certificates.
- **LOP and (RA or MA):** The LOP is supposed to hide the location from the RA and the MA, respectively, and may not be combined with any of these components.
- **SCMS Manager:** This component issues policies and rules defining the behavior of all the components of the SCMS. It is advisable to run it completely independently.
- **MA and (RA, LA, or PCA):** The MA should not be combined with any of the RA, the LA or the PCA. If combined, the MA could circumvent protocols which have to be in place for the MA to request information on the linkage of a set of certificates.
- **Root CA:** The root CA should be run separately and by a very trustworthy and well-monitored organization.

VI. Simplified System Models

This section presents potential simplifications of the SCMS.

A. Remove Intermediate CA

This shortens the chain of trust and reduces the number of CAs to be run at the cost of reducing flexibility and potentially increasing the Root CA's vulnerability to attacks, as the root CA will now be active and online for more time.

B. Remove RC

In order to remove the RC, the system may have only one RA or one has to make sure that each device can access only one RA. The main benefit of this simplification is that the SCMS structure is simpler to implement. The restriction to one RA makes it more difficult for a service provider to enter the system running a new RA.

C. Remove LOP

Removing the LOP simplifies the implementation further but also has a drawback. Both the RA and the MA can now observe the IP address of the requesting device. For the MA, the shuffling functionality of the LOP is also gone. There needs to be processes and practices in place so that these authorities do not abuse that knowledge.

D. Use Hardware Security Modules to Merge LAs

The SCMS is significantly simplified by merging the two LAs into one. Two separate LAs prohibit the operators from linking certificates. The proposed simplification relies on the idea to use a Hardware Security Module (HSM) [15], which limits the access of the operator to data on the device. This simplifies the design and lessens the requirement for computational power, and it has positive effects on the size of the CRL and the computational power required at the device for revocation purposes. Public perceptions of this approach would need to be weighed alongside the technical and business advantages of this simplification.

E. System Not Using CRLs

The last simplification is to abstain from CRLs. In this case, revocation is handled only by means of the RA's internal blacklist. Not using CRLs allows for removing the LAs completely and simplifying the MA by removing the CRLG as well as the CRLS and the CRLB. The resulting system is attractive due to its simplicity. However, in order for revocation by internal blacklisting to be effective, the super-batch of pseudonym certificates need to have a short validity period such that the devices need to request certificates frequently. This, again, requires the presence of frequent two-way communication.

VII. Conclusions

We introduced a Security Credential Management System for V2V communications that is currently the leading candidate design for the V2V security backend design in the US, subject to review by USDOT and other stakeholders. One of the main challenges is to find a right balance among security, privacy, and efficiency for a possibly mandated system. The proposed solution uses pseudonym certificates to sign messages and additionally, in the backend, separation of duties to provide a good trade-off between security and privacy. Follow-up studies include a cost and performance analysis, based on which a refined system will be designed and specified. More details on the SCMS can be found in [16].

Acknowledgements

The presented results are a culmination of efforts by many parties and people. This includes members of the US Department of Transportation (USDOT), the Crash Avoidance Metrics Partnership Vehicle Safety Consortium (CAMP VSC3), and the Vehicle Infrastructure Integration Consortium (VIIC). Funding was provided by the USDOT and OEMs participating in CAMP VSC3. The OEMs Daimler, Ford, GM, Honda, Hyundai-Kia, Nissan, Toyota, and VW-Audi participate in CAMP VSC3 and the VIIC. BMW and Chrysler participate in the VIIC. The authors are deeply grateful to Bill Anderson, Stephen Farrell, Scott Vanstone, and members of the USDOT and CAMP VSC3 for reviewing the manuscript and providing important suggestions for improvements.

References

[1] U.S. Department of Transportation - Research and Innovative Technology Administration. Vehicle-to-Vehicle (V2V) Communications for Safety. [Online]. Available: www.its.dot.gov/research/v2v.htm

[2] N. Bißmeyer, H. Stübing, E. Schoch, S. Götz, J. P. Stolz, and B. Lonc, "A generic public key infrastructure for securing Car-to-X communication," in *18th World Congress on Intelligent Transport Systems*, 2011.

[3] ETSI. TR 102 893V1.1.1 (2010-03) Intelligent Transport Systems (ITS); Security; Threat, Vulnerabilty and Risk Analysis (TVRA).

[4] ——. TS 102 731V1.1.1 (2010-09) Intelligent Transport Systems (ITS); Security; Security Services and Architecture.

[5] ——. TS 102 867 v1.1.1, Intelligent Transportation Systems (ITS); Security; Stage 3 mapping for IEEE 1609.2.

[6] Secure Vehicle Communication. Security Architecture and Mechanisms for V2V/V2I. [Online]. Available: www.sevecom.org/Deliverables/Sevecom_Deliverable_D2.1_v3.0.pdf

[7] Vehicle Safety Communications Consortium. Appendix H: Final Report 2006.

[8] 1609.2. Annex E.4.1: Why sign data instead of using a message authentication code? [Online]. Available: standards.ieee.org/findstds/standard/1609.2-2013.html

[9] U.S. Department of Transportation. Safety Pilot Model Deployment. [Online]. Available: safetypilot.umtri.umich.edu/

[10] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *MOBICOM*, 2008, pp. 116–127.

[11] J. R. Douceur, "The sybil attack," in *IPTPS*, 2002, pp. 251–260.

[12] IEEE. (2000) Std 1363-2000, standard specifications for public-key cryptography. [Online]. Available: ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=891000&contentType=Standards

[13] National Institute of Standards and Technology. (1999, July) Recommended elliptic curves for federal government use. [Online]. Available: csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.doc

[14] SEC, *Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV)*, SEC Std. 4, 2011.

[15] Wikipedia. Hardware security module. [Online]. Available: en.wikipedia.org/wiki/Hardware_security_module

[16] Anonymous. Defensive publication. [Online]. Available: priorart-database.com/pubView/IPCOM000210877D