A preliminary version of this paper appears in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2017*, ACM Press, 2017. https://doi.org/10.1145/3098243.3098257. This is the full version.

Binary Hash Tree based Certificate Access Management[*]

Virendra Kumar[†] Jonathan Petit[‡] William Whyte[§]

Abstract

We present a certificate access management system to support the USDOT's proposed rule on Vehicle-to-Vehicle (V2V) communications, Federal Motor Vehicle Safety Standard (FMVSS) No. 150. Our proposal, which we call Binary Hash Tree based Certificate Access Management (BCAM) eliminates the need for vehicles to have bidirectional connectivity with the Security Credential Management System (SCMS) for certificate update. BCAM significantly improves the ability of the SCMS to manage large-scale software and/or hardware compromise events. Vehicles are provisioned at the start of their lifetime with all the certificates they will need. However, certificates and corresponding private key reconstruction values are provided to the vehicle encrypted, and the keys to decrypt them are only made available to the vehicles shortly before the start of the validity periods of those certificates. Vehicles that are compromised can be effectively removed from the V2V system by preventing them from decrypting the certificates. We demonstrate that the system is feasible with a broadcast channel for decryption keys and other revocation information, even if that channel has a relatively low capacity.

Keywords: Binary tree, certificate, access management, revocation, connected vehicles, SCMS.

[*]This work is supported in part by Sirius XM Radio Inc., 1221 Avenue of Americas, New York, NY, USA. http://www.siriusxm.com.

[†]OnBoard Security, Inc., 187 Ballardvale St A202, Wilmington, MA, USA. vkumar@onboardsecurity.com.

[‡]OnBoard Security, Inc., 187 Ballardvale St A202, Wilmington, MA, USA. jpetit@onboardsecurity.com.

[§]OnBoard Security, Inc., 187 Ballardvale St A202, Wilmington, MA, USA. wwhyte@onboardsecurity.com.

Contents

1 Introduction

On January 12, 2017, the National Highway Traffic Safety Administration (NHTSA), U.S. Department of Transportation (USDOT) proposed to issue a new Federal Motor Vehicle Safety Standard (FMVSS) No. 150 [1], to require all new light vehicles to be capable of Vehicle-to-Vehicle (V2V) communications, such that they will send and receive Basic Safety Messages (BSMs) to and from other vehicles. In their Notice of Proposed Rulemaking (NPRM), NHTSA estimated that in year 30 of deployment, V2V communications will help prevent about half a million crashes and save about a thousand lives (cf. Table I-1 of [1]). Therefore, the security of V2V communications is paramount.

V2V system security largely depends on digital signatures [7, 20, 13], and in particular on vehicles having access to digital certificates to sign their BSMs. In provisioning vehicles with certificates, a balance must be struck: on one hand, if vehicles have fewer certificates than they need for the rest of their lifetime, they must periodically connect to a Public-Key Infrastructure (PKI) to download more certificates; on the other hand, the more certificates a vehicle has, the longer it has to stay on the Certificate Revocation List (CRL), if it is compromised.

The current approach considered for national deployment is specified in the IEEE Std 1609.2 [12], the PKI designed by Crash Avoidance Metrics Partners LLC (CAMP) called the Security Credential Management System (SCMS) [23], and the requirements and interfaces produced also by CAMP [14, 15]. In this approach, vehicles are provided with 3 years' worth of certificates (and the corresponding private key reconstruction values[1] in local storage and they can periodically "top up" back to 3 years' worth via a *bidirectional* connection with the SCMS. Also, CAMP estimates that the most constrained vehicles will have space to store only about 10,000 CRL entries, which means that the system can handle a *revocation rate*[2] of only 3,000.

Providing the bidirectional connectivity for certificate top-up is a significant contributor to the overall cost of the system [1]. Yet, the vehicles cannot be provisioned with more than 3 years' worth of certificates, as that would further reduce the revocation rate that the system can handle in the current design.

Contributions Our approach, called **B**inary Hash Tree based **C**ertificate **A**ccess **M**anagement (BCAM), builds upon the preliminary work of [18]. [18] is focused mainly on the different use cases of satellite communication, and the protocols are presented at a very high level. This work has new results and provides a thorough analysis along with experimental results.

In BCAM, vehicles are provisioned at the start of their lifetime with all the certificates they will need, and as in the current system, the corresponding private keys are generated on (or, securely injected into) the vehicle such that they are only known to the vehicle and not to any other entity in the system. However, when the certificates (and the corresponding private key reconstruction values) are provided to the vehicle, they are *encrypted*, and the keys to decrypt them are only made available to the vehicles shortly before the start of the validity periods of those certificates. Note that the pseudonym certificate model (i.e., certificate validity, number of concurrently valid certificates, etc.) is unchanged and exactly like the current system designed by CAMP. This in turn

[1]Reconstruction values are used in implicit certificates [4], which are a variant of public key certificate, such that a public key can be reconstructed from any implicit certificate, and is said then to be *implicitly* verified, in the sense that the only party who can know the associated private key is the party identified in the implicit certificate. The issuing Certificate Authority also provides a private key reconstruction value for the receiver to reconstruct its private key for signing.

[2]Revocation rate is the number of vehicles revoked per year.

means that the SCMS use cases of misbehavior reporting and global misbehavior detection are also unaffected by the BCAM system and should work exactly like the current system. Revocation is modified as follows: vehicles that are compromised are prevented from participating in the V2V system by not giving them the necessary keys for decrypting the certificates (and the corresponding private key reconstruction values).

The benefits of BCAM are twofold: (1) two-way communication with the SCMS for certificate download is eliminated except for extremely long-lived vehicles or vehicles that need for some reason to be re-initialized; and (2) since revocation is enforced on the send-side, if vehicles are revoked due to misbehavior or malfunction, they can be prevented from sending valid signed messages altogether. Enforcing revocation on the send side eliminates the need for checking revocation as is the case with any CRL-based system, and allows the system to handle widespread compromise much more robustly and effectively than the current SCMS design.

The SCMS diagram, amended to support BCAM, is shown in Figure 1. Compared to the existing SCMS design, a new SCMS component called Certificate Access Manager (CAM) is added, and two existing SCMS components related to Certificate Revocation List (CRL) are renamed to reflect their new roles.

1. CRL Generator $\Rightarrow$ Revocation Generator (RG)

2. CRL Broadcast $\Rightarrow$ Certificate Access Broadcast (CAB)

In this paper, we show that it is practical to broadcast the decryption information by significantly compressing it using binary hash trees [17]. We present a novel encoding/decoding algorithm for complete[3] binary trees. We are unaware of any published encoding approaches for complete binary trees which in general produce smaller encodings than ours while keeping the decoding time linear in the depth of the tree. Our decoding algorithm, which is a breadth-first algorithm, is also fast because the max queue size in our algorithm is the number of revoked vehicles and not the full breadth of the tree.

We show that a 64 kbps broadcast channel is sufficient to guarantee that vehicles will receive the decryption keys (and other revocation information), if they are entitled to them, in a reasonable amount of time, such as a couple of hours of driving[4], unless the number of compromised vehicles is unrealistically high. Appropriate network coding, such as LT Codes [16] or Raptor Codes [22], is strongly recommended to be used with the broadcast channel, so that the vehicles can seamlessly download the broadcasted material over multiple sessions without any unnecessary overheads.

Outline The remainder of this paper is structured as follows. Section 2 presents related work. Our design goals and an overview of the system are presented in Section 3. Section 4 details the BCAM provisioning. Section 5 provides a discussion of security and other topics that arise from this proposal. Finally, Section 6 summarizes this paper.

2 Related Work

This section discusses related work other than [18], whose relation to this work was already discussed in the Introduction.

[3]In a complete binary tree every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible.

[4]On average, Americans spent 46 minutes per day behind the wheel in 2013. http://newsroom.aaa.com/2015/04/new-study-reveals-much-motorists-drive/.

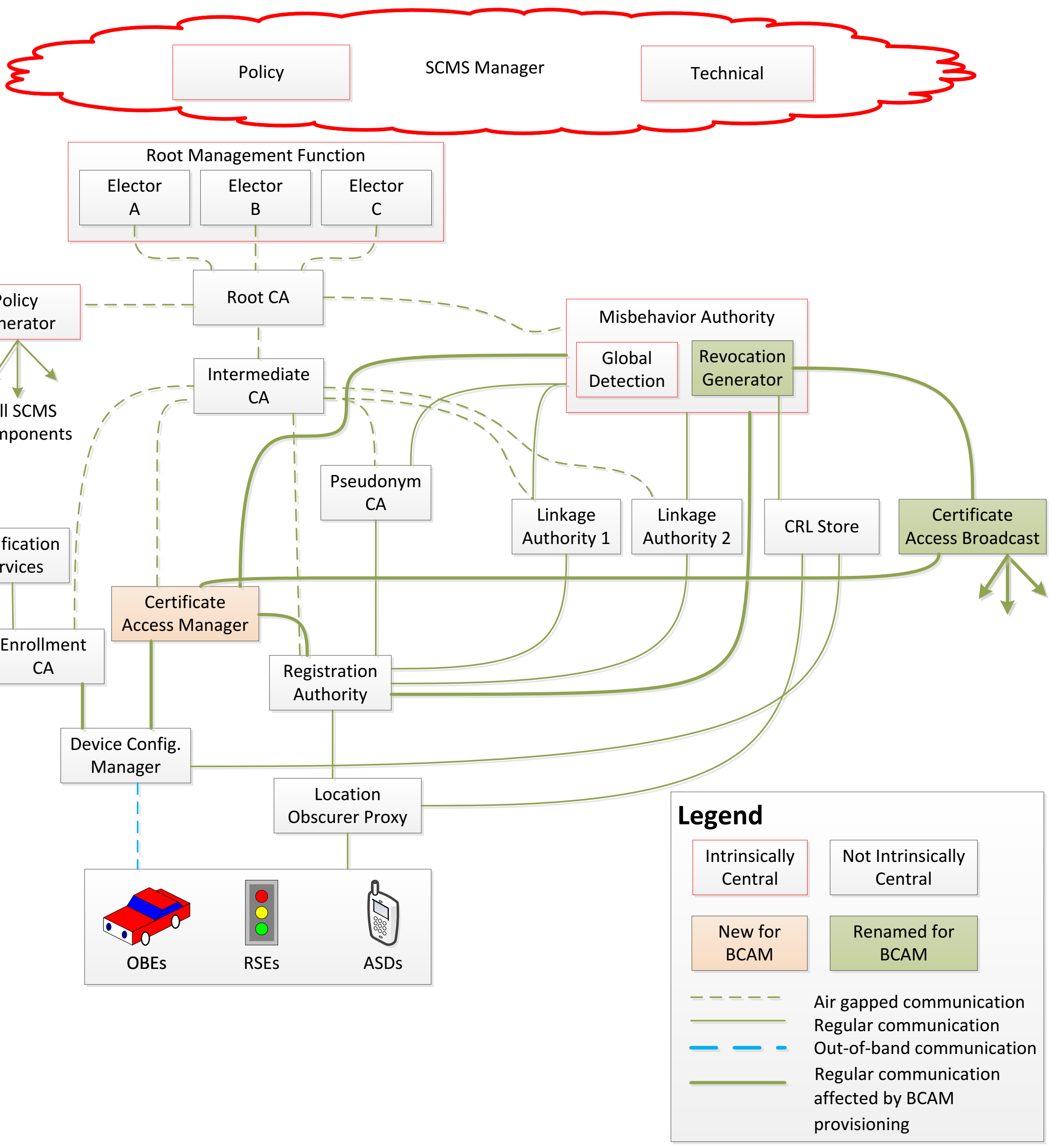

Figure 1: SCMS including Certificate Access Manager

The concept of Broadcast Encryption (BE) was introduced by Fiat and Naor in [8]. In this paradigm, a broadcaster encrypts messages and transmits them to a group of users $\mathcal{U}$ who are listening to a broadcast channel and use their private keys to decrypt transmissions. The broadcaster may exclude any subset of users $\mathcal{R} \subseteq \mathcal{U}$ from being able to decrypt the contents of the broadcast using a one-time exclusion or revocation mechanism [6]. The solutions to BE, however, do not directly translate into solutions for our problem, because a non-revoked user can help a revoked user to gain access to the sensitive information being broadcast (since this information is the same for all users). In BCAM, a revoked vehicle (or, an adversary that has extracted its private keys, encrypted certificates, etc.) should not be able to decrypt its certificates even if it colludes with any number of non-revoked vehicles.

[10] proposed a Bandwidth Efficient Certificate Status Information (BECSI) mechanism to efficiently distribute certificate status information (CSI) in vehicular ad hoc networks. By means of Merkle hash trees (MHT), BECSI allows to retrieve authenticated CSI not only from the infrastructure but also from vehicles acting as mobile repositories. Since these MHTs are significantly smaller than the CRLs, BECSI reduces the load on the CSI repositories and improves the response time for the vehicles.

In [11], authors presented an Efficient and Privacy-Aware (EPA) revocation mechanism that relies on the use of positive proofs of the certificate's no-invalidity instead of forcing vehicles to download huge revocation lists. A no-invalidity proof gives evidences that a given certificate has not been revoked. These proofs are obtained from a MHT that is constructed from the list of revoked certificates.

However, none of these solutions eliminate the need of bidirectional communication, and still put the burden on all valid vehicles instead of performing revocation *at source* as proposed in this paper.

3 Overview

3.1 Design Goals

The BCAM approach has the following high-level goals:

- Remove the need for bidirectional connectivity to the SCMS for certificate provisioning.
- Provide a way to completely remove misbehaving / malfunctioning vehicles from the system by removing their certificates and keys.
- Provide a mechanism for reinstating vehicles that were wrongly revoked, or which were revoked due to some fixable error which has been fixed.

In addition to the above, BCAM has the following lower level goals:

- Use HSMs to provide highly trusted services to vehicles.
- Keep the complexity of HSM implementation as low as possible.

3.2 Terminology

In the Binary Hash Tree based Certificate Access Management (BCAM) provisioning, vehicles are provisioned with a lifetime worth of certificates, say 30 years[5], that are encrypted and stored in standard, non-secure persistent storage. As with the current CAMP design, there are a number of certificates simultaneously valid for a given period of time. The period of time is referred to as the *certificate validity period* or the *time window*. It is currently set to a week in CAMP, and this need not change in the BCAM approach. The collection of certs simultaneously valid within a time window is referred to as a *batch*, and the number of certs in a batch can be baselined at 20 as in the current CAMP design. In addition to the existing time window equal to the certificate validity period, there is a time window, equal to or an integer multiple of the certificate validity period, which is referred to as the *BCAM period*, and which can be encoded as a 16-bit value. A vehicle has one or more batches of certs within a given BCAM period, depending on how many certificate validity periods fit into a BCAM period. All of the vehicle's certificates within a single BCAM period are consolidated into a single *BCAM batch*; the BCAM batch is encrypted with a (vehicle and period)-specific symmetric key, the *BCAM batch key*, that is initially known only to the Certificate Access Manager (CAM) in the SCMS and revealed at the appropriate time to the vehicle's Hardware Security Module (HSM). The length of time that all the BCAM batches of a vehicle spans is called the *BCAM lifetime*.

3.3 General Approach

In BCAM, access to certificates and revocation are managed using three mechanisms.

3.3.1 "Soft" Revocation List

This is the first line of defense against compromised vehicles. Every BCAM period, the system distributes a "soft" list of revoked vehicles in the group, referred to as a Soft Revocation List (SRL). This list is signed by the Revocation Generator and is checked by a dedicated Hardware Security Module (HSM) on the vehicle. If the vehicle is on the SRL, the HSM will refuse to recover the decryption key, even though in principle such recovery would be possible, and the vehicle will not have access to its certificates going forward.

This mechanism is sufficient to withdraw from service any vehicle that is subject to a compromise of the software but not the HSM. We believe software compromise to be significantly more likely than hardware compromise, so SRL will remove the vast majority of compromised vehicles from the system.

A soft-revoked vehicle can be reinstated by removing its identifier from the SRL. Note that the SRL is used once per BCAM period, by the vehicle's HSM at the time it decrypts certificates, rather than needing to be checked every time a message is received as is the case with CRLs. This means the SRL itself does not need to be stored persistently; only the results of the HSM's check on its host vehicle's validity need to be stored.

3.3.2 Standard CRL

If a vehicle is on the SRL, but is also active and using valid certificates, it is a clear indication that not only the software but the vehicle's HSM is also compromised. Such vehicles are put on

[5]The current CAMP design has enrollment certificates with validity period of 30 years, that are intended to last the lifetime of a vehicle.

the standard CRL, which can also be distributed through the broadcast channel. At any time, the BCAM system allows devices access to their certificates for the current and the next BCAM period. So, devices will rarely need to stay on the standard CRL for longer than one broadcast distribution cycle (typically a week) as their removal will in general be managed by withdrawing access to their certificate decryption keys. Thus, CRLs are not essential and can be completely done away with in the BCAM system, if a 1-week delay in removing HSM-compromised devices were deemed acceptable.

3.3.3 Device-Specific Values and "Hard" Revocation

Vehicles whose HSMs are believed to be compromised are removed from the system by preventing them from decrypting new certificates by a mechanism we call hard revocation. Every BCAM period, the system uses the broadcast channel to distribute device-specific values (DSVs), which are required to recover the decryption keys needed for decrypting the certificates. By the use of binary hash trees [17], the amount of data needed to distribute the DSVs to say 350 million vehicles[6] can be made to scale slightly sub-linearly in the number of hard-revoked vehicles rather than linearly in the total number of vehicles in the system. This makes distribution of individual DSVs practical, especially given that the number of hard-revoked vehicles is the number of vehicles that have had a compromise not only of the software but also of the HSM, and so that number can be assumed to be relatively small.

A hard-revoked vehicle can be reinstated by starting to broadcast its DSV again. (A hard-revoked vehicle will also be soft-revoked, so it will also be necessary to remove its identifier from the SRL). Also, note that just like the SRL, DSVs are used once per BCAM period, by the vehicle's HSM at the time it decrypts certificates, rather than needing to be checked every time a message is received as is the case with CRLs.

3.4 SCMS Architecture

In BCAM provisioning, certificate generation is carried out by the Pseudonym Certificate Authority (PCA) and Registration Authority (RA) as in the current CAMP design, but certificates are pre-generated for the entire life of a vehicle, i.e. 30 years, as opposed to just 3 years in the current design. A new component called Certificate Access Manager (CAM) is added to the SCMS for BCAM provisioning:

3.4.1 Certificate Access Manager

A component that is involved in managing devices' ability to access their certificates and private keys. It:

- Generates BCAM batch keys and uses them to encrypt the BCAM batches of certificates (In BCAM provisioning, encryption of BCAM batches by the CAM is done in addition to the encryption of individual pseudonym certificates done by the PCA in the current SCMS design.)
- In order to generate the BCAM batch keys, generates Device-Specific Values (DSVs) which can be used by the devices to derive the corresponding BCAM batch keys

[6]The Bureau of Transportation Statistics, USDOT reported that there were about 260 million registered vehicles in 2014 in the U.S.A., so one can safely assume that the number of active light-duty vehicles shall not exceed 350 million in the foreseeable future. `https://www.rita.dot.gov/bts/sites/rita.dot.gov.bts/files/publications/national_transportation_statistics/html/table_01_11.html`

There may be more than one CAM in the system (i.e., in SCMS terminology it is not "intrinsically central").

The following existing SCMS components have been renamed (and functionalities added to them) for BCAM provisioning:

- **CRL Generator $\Rightarrow$ Revocation Generator (RG)**: In addition to signing CRLs, RG is responsible for also signing Soft Revocation Lists (SRLs) every BCAM period.
- **CRL Broadcast $\Rightarrow$ Certificate Access Broadcast (CAB)**: In addition to broadcasting CRLs, CAB is responsible for also broadcasting SRLs (generated by RG) and DSVs (generated by CAM) every BCAM period.

Additionally, the RA's functionality is extended to include interactions with the CAM to create the encrypted BCAM batches.

These new and renamed components, and the new RA-CAM interface, are illustrated in Figure 1.

3.5 Certificate Access Messages

Prior to the start of each BCAM period, the CAB broadcasts update messages for all the vehicles in the system that control the vehicles' access to their certificates for that BCAM period. These updates consist of SRL, DSVs, and CRL.

3.5.1 SRL

A list of vehicles whose software are believed to be compromised but whose HSMs are still believed to be secure. SRLs are signed by the RG and then compressed. The BCAM provisioning system requires that the signature on SRL (including the RG's certificate and its chain up to the Root CA) be verified inside the vehicle's HSM. See 4.5.1 for more details on SRL, and 5.4.3 for a discussion on the security of SRLs. (As the HSM of the vehicle on the SRL is secure, it is assumed to enforce SRL by not decrypting the BCAM batch even though the corresponding DSVs needed for deriving the BCAM batch keys are broadcast by the CAB.)

3.5.2 DSVs

A set of values that can be used by any vehicle that is not hard-revoked to derive the BCAM batch key for decrypting the BCAM batch. If a vehicle's HSM is compromised, it can no longer be trusted to follow the protocols, and thus needs to be removed from the system. This is accomplished by no longer sending its DSVs. The DSVs are compressed using a binary tree approach, but even so, revocation by this approach creates relatively large packets per revoked device, which is why it is only used if the HSM is believed to be compromised. This list is signed by the CAM, but need not be verified within the HSM; see discussion in 5.4.1.

3.5.3 CRL (Optional)

The format and frequency of CRL have been specified in the existing SCMS specifications. The CRL contains revocation information for devices that are on the SRL but are active and using valid certificates decrypted from the BCAM batch, indicating that not only the software but the vehicles' HSMs are also compromised. However, a device needs to stay on the CRL only until the end of next BCAM period, when it will be hard-revoked (and thus won't be able to decrypt BCAM batches to obtain new certificates) by not sending its DSV.

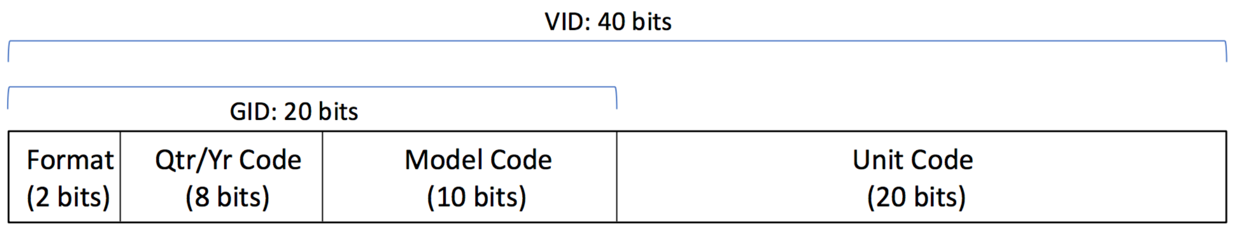

Figure 2: VID and GID

4 BCAM Details

4.1 System Setup

BCAM batch keys are derived from a vehicle-generated key (VK), a CAM-generated Device Specific Value (DSV), and the BCAM period t, which the HSM must ensure is equal to the BCAM period t' in the SRL. A specific VK is denoted by vk. A specific DSV for the vehicle with unique identifier id in BCAM period t is denoted $\delta_{id,t}$.

4.1.1 Vehicle Identification

Each vehicle in the system has a unique 40-bit Vehicle Identifier (VID) constructed as illustrated in Figure 2. It is comprised of a 2-bit Format field, an 8-bit Quarter/Year Code field, a 10-bit Model Code field and a 20-bit Unit Code field. The Group Identifier (GID) is defined as the 20 most significant bits of the VID. SRL is divided into groups to keep the size of SRL from growing indefinitely.

4.1.2 Cryptographic Primitives

Below we define the different cryptographic primitives used in BCAM provisioning. These definitions need to be public as they will be used by vehicles to derive their BCAM batch keys. In what follows, for a bit b and an integer k, b^k indicates a bit string containing the bit b repeated k times; and for bit strings x and y, $x\|y$ denotes their concatenation. $\mathcal{H}$ is a collision-resistant hash function, such as the Secure Hash Standard SHA-256 [19]. $\mathcal{M}$ is a secure message authentication code, such as the SHA-256 hash-based HMAC [9].

- Left and right hash functions, denoted $\mathcal{L}$ and $\mathcal{R}$, on input: binary tree node value δ and padding length l
$$\mathcal{L}(\delta) = \mathcal{H}\left(\delta\|0^l\right), \ \mathcal{R}(\delta) = \mathcal{H}\left(\delta\|1^l\right) \tag{1}$$
- BCAM batch key derivation function, denoted $\mathcal{K}$, on inputs: vehicle-generated key vk, vehicle identifier id, BCAM period t, CAM-generated device-specific value δ
$$\mathcal{K}(vk, id, t, \delta) = \mathcal{M}\left(vk, \mathcal{M}(vk, id)\|\mathcal{M}(vk, t)\|\mathcal{M}(vk, \delta)\right) \tag{2}$$

Remarks:

1. For the specific choice of SHA-256 for $\mathcal{H}$, we recommend $l = 192$. This is because the SHA-256 compression function takes input that is 512 bits long, and SHA-256 appends the final block of input with a 64-bit length indicator, so an input of length 256 bits (for δ) + 192 bits

(for the padding) = 448 = 512 - 64 (for the length indicator) is the maximum size for one invocation of the compression function.

2. The reason for the nested construction of BCAM batch key derivation function is that it provides protection against changes in the length of the input values in the future; if id, t, and δ were concatenated without any length indication and the lengths changed, it is conceivable that one pre-change $(id\|t\|\delta)$ combination could match a post-change combination. Likewise, an encoding that explicitly indicates the length, while it would be secure against changes in length as such, might be vulnerable to a future decision to change the encoding approach. We believe the nested construction to be the most future-proof.

3. The BCAM period t used by the HSM must come from the SRL. See Section 5.4.3 for rationale.

4.1.3 Device-Specific Values

Device-Specific Values (DSVs) are calculated using a unique binary hash tree for each BCAM period. For efficiency reasons, it is recommended that each CAM uses a single binary tree for all the vehicles it serves. DSVs, which are the leaf node values of a binary hash tree, are generated as follows:

1. Each CAM, for each BCAM period t, selects a 256-bit random seed value, denoted δ_t.

   (It is important that seed values for two different CAMs or BCAM periods are unrelated so that revealing the seed value of a given CAM and BCAM period doesn't reveal anything about any other seed value of either a different CAM or a different BCAM period.)

2. For each vehicle with a unique identifier id, calculates DSV $\delta_{id,t}$ via a binary hash tree as follows:

   (a) Set $\delta_{id,t} = \delta_t$

   (b) For each bit b in the binary representation of id, starting from the leftmost bit:

      i. If $b = 0$, set $\delta_{id,t} = \mathcal{L}(\delta_{id,t})$

      ii. If $b = 1$, set $\delta_{id,t} = \mathcal{R}(\delta_{id,t})$

   (c) Output the final $\delta_{id,t}$.

The binary tree provides a means to compress information about DSVs. Indeed, a node in the tree can be used to derive values of all its child (and grandchild, and so on) nodes. This means that if any node value between a vehicle's leaf node and the root node is made public, the vehicle can derive its DSV from that node's value by using the appropriate combination of left and right hashes. This is illustrated in Figure 3. In the upper part of Figure 3, no node is revoked, and so the root node is made public and all leaf nodes can be derived from it.

In the lower part of Figure 3, node 100 is revoked, and so the rest of the keys are made public by revealing the following node values, which in the diagram are heavy-outlined:

- Node 0 (enables the derivation of leaf nodes 000, 001, 010, 011)
- Node 11 (enables the derivation of leaf nodes 110, 111)
- Node 101

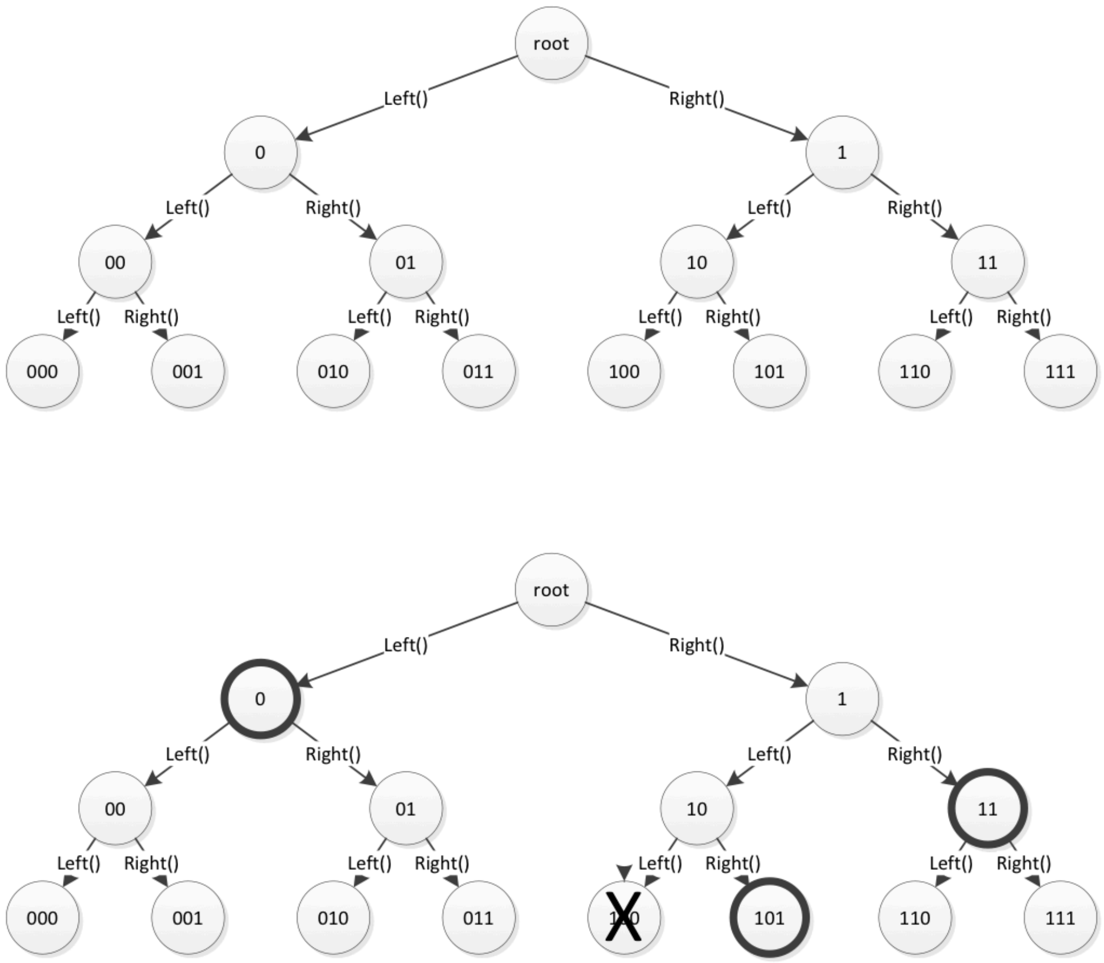

Figure 3: Binary Trees

For a system with a total of n users out of which r users are revoked, the broadcast message will on average include $\left(r \cdot \log_2 \left(\frac{n}{r}\right)\right)$ number of binary tree nodes (cf. Theorem 1 of [2]) for $1 \le r \le n/2$. A proposal for encoding of node positions in the binary tree is presented in Section 4.4.

The use of the DSVs to generate the BCAM batch keys for encrypting/decrypting BCAM batches is described in Section 4.2. The use of the binary trees to remove vehicles from the system is described in Section 4.4.

4.1.4 Initialization and Configuration

The SCMS components involved are the CAM, the SCMS Manager, and the vehicle Configuration Managers (DCMs). In the SCMS design, the DCMs have responsibility for provisioning the vehicle with trust management information, for example root CA certificates and the URL of the RA, and also for attesting to the Enrollment CA that the vehicle currently applying for certificates is entitled to them. Each CAM is responsible for a set of vehicles. When a new CAM is created, it communicates with the SCMS Manager to get the initialization parameters, which include a unique identifier, hash function to be used, CA info for getting a certificate, etc.

On initialization, and every quarter thereafter, the CAM communicates with the SCMS Manager to get information, e.g., a list of vehicles it is responsible for, their DCMs, their lifetimes, encryption algorithm to be used, etc.

4.2 Certificate Provisioning

A vehicle's On-Board Equipment (OBE) is considered to have two parts to its architecture: a host processor and an HSM. This is illustrated in Figure 4. Applications that can access the cryptographic keys without human authentication, known as *privileged applications*, run on the host processor. Cryptographic keys themselves are stored within the HSM. The keys can be used by processes outside the HSM, subject to access control mechanisms enforced by the host processor, but never leave the host processor in plaintext. This architecture is already accepted in the Connected Vehicle setting, but this proposal requires additions to functionality typically required of HSMs in the Connected Vehicle setting, e.g., it must be able to export the vehicle key vk encrypted to the CAM, it must be able to parse and verify signature on the SRL, it must be able to vehicle BCAM batch keys, etc.

Both the Host Processor and the HSM are required to be secure. However, this proposal assumes that the Host Processor cannot be made as secure as the HSM, so the probability of Host Processor compromise is significantly greater than the probability of HSM compromise. This is a reasonable assumption: The Host Processor runs more, and more complex, software than the HSM does, and the Host Processor software is more likely to be frequently updated. Note that following from this assumption we make certain design recommendations based on the principle that the software inside the HSM should be kept as limited as possible as noted in the design goals in Section 3.

Certificate provisioning is expected to follow these steps:

0. **Initialization (RA)**. The RA is provided with the length of the BCAM periods and the information about which CAM is responsible for which vehicles.
1. **Certificate Request**. The vehicle requests certificates from the RA just as it would have in the current SCMS, except it includes the encrypted (to the CAM) vehicle key vk in the request.
2. **Certificate Request Processing**. It differs from the current SCMS in the following ways:

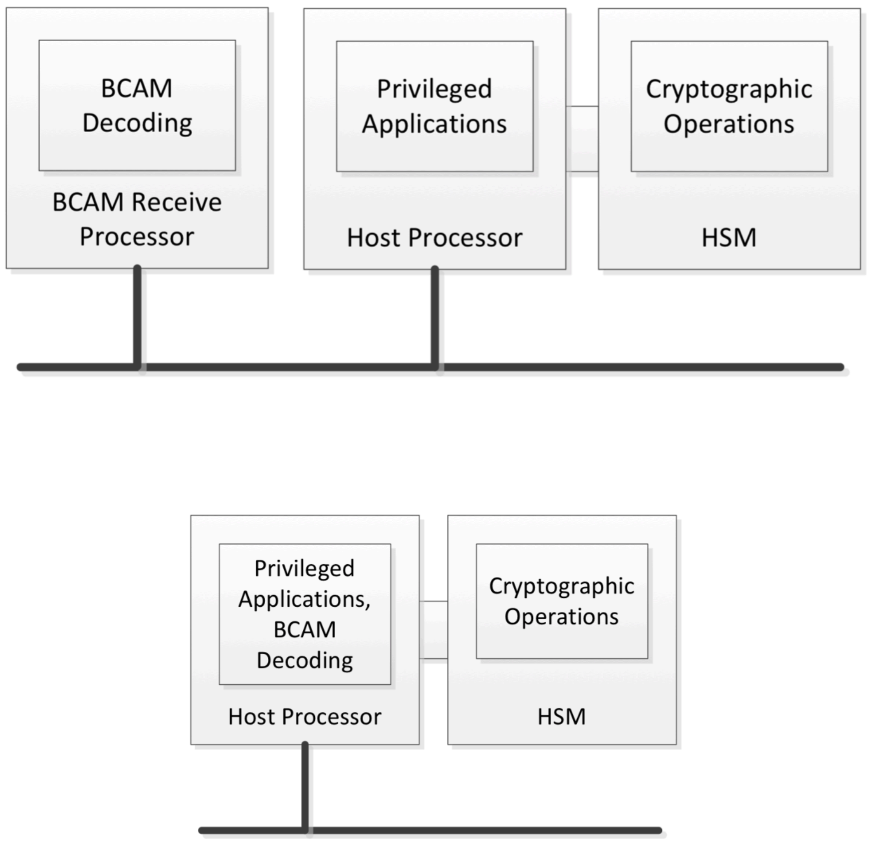

Figure 4: OBE Architecture

(a) The RA first determines if the vehicle needs to be provisioned with BCAM batches or not. If so, then divides all certificates into different BCAM periods to form (unencrypted) BCAM batches, and gives them along with the vehicle identifier id and the encrypted vk to the CAM.

(b) For each BCAM period t, the CAM generates the BCAM batch key $ek_{id,t}$ using the key derivation function defined in 4.1.2 with inputs $(vk, id, t, \delta_{id,t})$, where $\delta_{id,t}$ is the binary tree node value generated by the CAM for id and t.

(c) The CAM encrypts each BCAM batch with the corresponding $ek_{id,t}$, and gives all the encrypted BCAM batches to RA, which makes them available for download by the vehicle.

3. **Vehicle Processing**. The vehicle downloads the encrypted BCAM batches just as it would have downloaded the certificates in the current SCMS.

4.3 Operations

Every BCAM period, the Certificate Access Broadcast (CAB) broadcasts a Certificate Access List (CAL, named by analogy with Certificate Revocation List) containing all the information needed for legitimate vehicles to decrypt that BCAM period's and the following BCAM period's certificates.

The Certificate Access List contains two sections plus a header:

- **Header**: Contains date of issuance. This enables mid-week revocation, see 4.3.1.
- **Device-Specific Values List**: It contains CAM ID *cid*; BCAM period t; node values covering *all non-hard-revoked* vehicles for $t, t+1$: $\Delta_{cid}(t), \Delta_{cid}(t+1)$; and the CAM's signature.
- **Soft Revocation List (SRL)**: It contains BCAM period t; a bitmap of length equal to the number of vehicles in the system, where at any position u, a "1" indicates that the vehicle u has been soft-revoked, and "0" otherwise; and the Revocation Generator's signature.

Notes:

1. The SRL bitmap is expected to be sparse and can be significantly compressed using standard compression techniques. The DSV list, on the other hand, cannot be compressed as it contains cryptographically random data.

2. The SRL bitmap is signed-then-compressed (as opposed to compressed-then-signed), because the HSM has to verify the signature on the bitmap and check for the status of the OBE's bit within the bitmap. If it were compressed-then-signed, the HSM would have to also implement decompression, which runs contrary to our design goal to have as little functionality as possible inside the HSM.

On receiving the CAL, each correctly-behaving vehicle id with key vk takes the following steps:

1. Decompresses and verifies the signature on the SRL for its group, and aborts if the signature does not verify.

2. (Within the HSM) Parses the bitmap and aborts if id is identified as soft-revoked, otherwise stores the BCAM period t from the SRL and continues.

3. Verifies the signature on the DSV list and aborts if the signature does not verify, otherwise derives its DSV for BCAM periods $t, t+1$ (denoted $\delta_{id}(t), \delta_{id}(t+1)$, respectively) from $\Delta_{cid}(t), \Delta_{cid}(t+1)$, respectively.

4. (Within the HSM) Derives the BCAM batch keys for periods $t, t+1$ (denoted $ek_{id,t}, ek_{id,t+1}$, respectively) using $vk, id, t, \delta_{cid}(t), \delta_{cid}(t+1)$ using the derivation function defined in 4.1.2. Note that t is from the SRL in step 2.

5. Decrypts the certificates for periods $t, t+1$.

Steps 2 and 4 are carried out within the HSM on the OBE, i.e. the HSM has the responsibility for enforcing soft revocation via SRL. If the HSM is compromised, soft revocation will not work; in this case, hard revocation is enforced by removing the DSV for that vehicle from $\Delta_{cid}(t), \Delta_{cid}(t+1)$.

4.3.1 Mid-Week Revocation

In general, once an OBE has received the current CAL, it need not download anything more till the end of the next BCAM period. However, it is conceivable that an OBE needs to be revoked mid-week because it is observed to be misbehaving in a serious way. In this case the OBE is added to the SRL and the header information in the CAL is updated to indicate the generation time of the new SRL. Receiving OBEs can note that the CAL has a new generation time and run the receive-and-reconstruct process again. Additionally, if mid-week *hard* revocation is necessary, it can be implemented by updating the standard CRL.

4.4 Binary Tree Encoding

We propose to encode the tree and the DSV values separately: the encoding of the tree indicates which nodes are present, and is followed by a list of DSV values such that the number of DSV values is equal to the number of published nodes. The problem of encoding the DSV list thus becomes a problem of efficiently encoding the tree.

A standard textbook approach for encoding the topology of a *general* binary tree with n nodes that is not necessarily complete like the one being used here is to create a bit string of length n, with each bit set to 0 for a leaf node (i.e., a node with no children) and to 1 for an internal node (i.e. a node with 2 children). The binary trees for use in this setting are slightly different:

1. All branches of the original tree are of the same depth, so the topology is known; the only question is which nodes are published and which are omitted.

2. All published nodes allow the derivation of the full subtree below them, so once a node is known to be published there is no need to publish any of its child nodes.

In other words, nodes are in the state "omitted", "published", or "can be derived from published nodes", where only nodes in the "published" state need to have the corresponding value published, and the state of a node need only be given if its parent node is in the "omitted" state.

To examine encoding effectiveness, we consider three different scenarios (also see Section 4.5.2 for real-world experiments with binary tree of depth 40). On day 1 (Figure 5) no vehicle is revoked. On day 2 (Figure 6) vehicles 2, 3 and 7 are revoked. In the pathological case (Figure 7) every odd-numbered vehicle is revoked. In the figures, *striped* leaf nodes indicate revoked vehicles, striped nodes (both leaf and non-leaf) are the nodes on the path from revoked vehicles to the root node and hence must never be published; *solid black* nodes indicate the nodes published; and *uncolored* nodes indicate the nodes covered by solid black nodes.

4.4.1 Encoding Algorithm for the CAM

It takes as input the colored tree, and outputs an encoded bit string *encString* and a list of DSVs *dsvList*:

1. Start from the root node of the tree and with an empty bit string *encString* and an empty DSV list *dsvList*.
2. If the node is *not* present in the collection of subtrees, i.e. striped node, append 0 to *encString*; go to the next node to the right or, if at the rightmost node in a row, go to the first node on the next row.
3. If the node is present, i.e. solid black node:
   (a) Append 1 to the *encString*.
   (b) Add the node value to *dsvList*.
   (c) Go to the next node and return to step 2.
4. If the node is a child of a present node, i.e. uncolored node, do *not* append anything to *encString*; go to the next node and return to step 2.

The above steps can be summarized as: **0 for striped, 1 for solid black, and nothing for uncolored**.

4.4.2 Decoding Algorithm for the Vehicle

It takes as input a vehicle identifier *id*, and the outputs of the encoding algorithm *encString* and *dsvList*. It outputs a (binary tree) node's DSV and the node's depth in the tree:

1. Start with $curDepth = 0$, $curPosition = 0$, $bitsBefore = 0$, $bitsAfter = 0$, $nodesPresent = 0$.
2. Take the encoding of the current layer, i.e. $(bitsBefore + 1 + bitsAfter)$ bits in *encString* starting from *curPosition*.
   (a) Initially this is the single first bit of the encoding.
   (b) Note, in the top layer there's only one bit, which is always the bit of interest. So *bitsBefore* and *bitsAfter* are both 0.
3. Set $nodesPresent = nodesPresent +$ the number of 1s in the first *bitsBefore* bits.
4. The bit of interest is the one that comes after the first *bitsBefore* bits. If the bit of interest is 1, stop and output $(nodesPresent + 1)$th entry in *dsvList* and *curDepth*.
5. Otherwise:
   (a) Set $curPosition = curPosition + (bitsBefore + 1 + bitsAfter)$
   (b) Set $bitsBefore = 2\times$(the number of 0 bits in the first *bitsBefore* bits)
   (c) If the bit at the $(curDepth + 1)$th position of *id* is 1, add 1 to *bitsBefore*.
   (d) Set $nodesPresent = nodesPresent +$ the number of 1s in the trailing *bitsAfter* bits

(e) Set $bitsAfter = 2\times$ (the number of 0 bits in the $bitsAfter$ bits after the bit of interest)

(f) If the bit at the $(curDepth + 1)$th position of id is 0, add 1 to $bitsAfter$.

(g) Set $curDepth = curDepth + 1$ and return to step 2.

Now, when applied to the three scenarios.

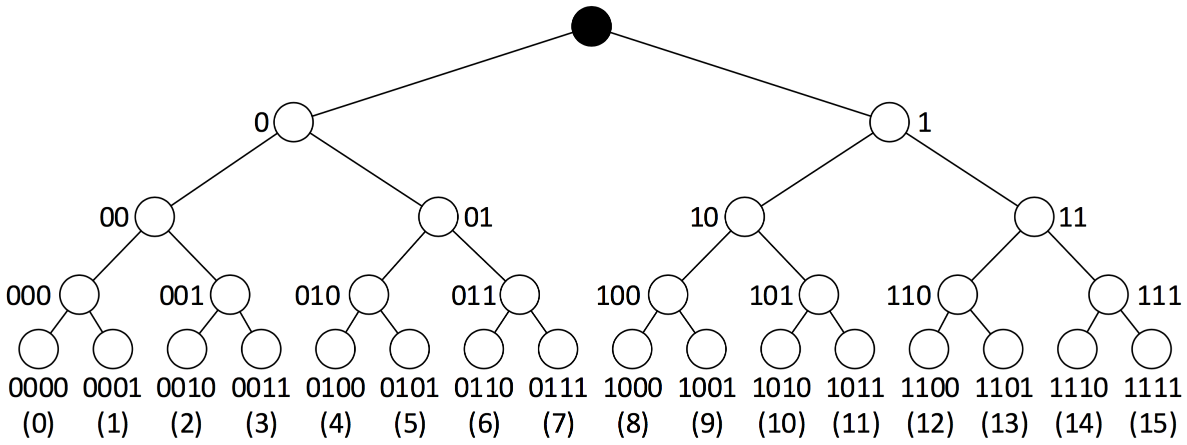

Figure 5: Day 1

1. Day 1 with no revoked vehicles, gets encoded as 1.

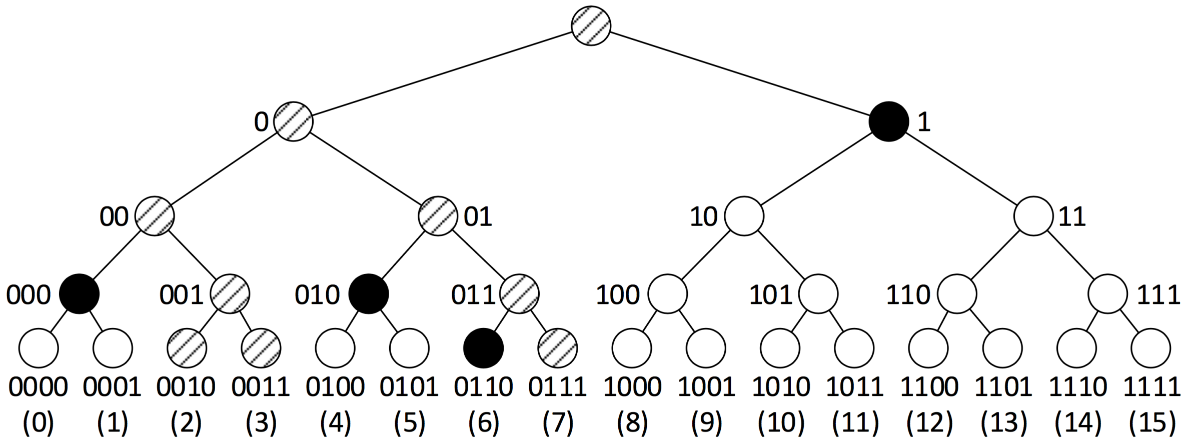

Figure 6: Day 2

2. Day 2 with 2, 3, 7 revoked, gets encoded as 0 01 00 1010 0010 (= 13 bits).

3. Pathological with every other vehicle revoked, gets encoded as 0 00 0000 00000000 1010101010101010 (= 31 bits).

For a binary tree of depth d, on an average, our encoding algorithm produces encodings that are $\frac{d}{2}$ times smaller in size than the basic encoding algorithm (where each published node is uniquely

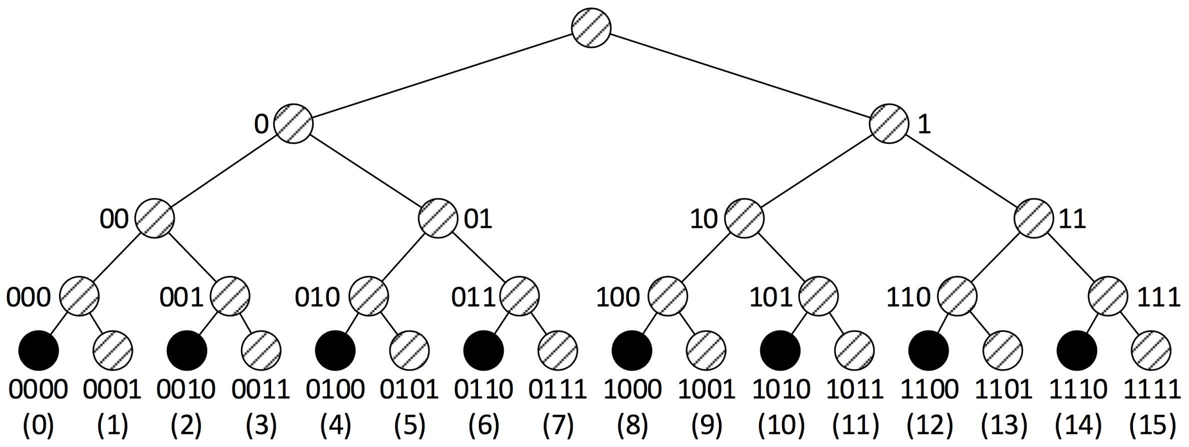

Figure 7: Pathological

represented using $d + 1$ bits). For the specific use case of SCMS where tree depth would be 40 so that there is a unique leaf node for every possible vehicle ID, our algorithm produces encodings that are *20 times* smaller in length. Note that the nodes to be published (i.e., solid black) are the siblings of the nodes that are marked revoked (i.e., striped). So, if the number of revoked nodes is much smaller than the total number of nodes, and if the revoked nodes are randomly distributed throughout the tree, it is very likely that only a small fraction of the sibling pairs in the tree will have both nodes marked revoked. Thus, the number of solid black nodes is roughly equal to the number of striped nodes, resulting in an encoding of size roughly 2 times the number of published nodes.

Our decoding algorithm, which is a breadth-first algorithm, is also fast, because the max queue size is the number of revoked vehicles. This is significant because the full breadth of our tree is 2^{40} compared to the number of revoked vehicles, which is expected to be in thousands.

4.5 Experimental Results

In this Section we discuss the bandwidth requirements for broadcasting SRL and DSV list.

4.5.1 SRL

SRL is a bitmap (i.e., 0=valid, 1=soft-revoked) in sequential vehicle ID order such that the revocation status is identified by bit position in the list. An uncompressed SRL is about 42 megabytes (MB) for 350 million cars, as illustrated below.

- 1,048,900 bits per Group of size
  - 20 bits: Group ID
  - 16 bits: BCAM period
  - 32 bits: compression parameters
  - bits: 1 bit per vehicle
  - 256 bits: signature

Table 1: Broadcast Latency for SRL

Soft-revoked (%)	Avg. entropy per bit	Compressed SRL size (MB)	Latency (min.)
1	0.08	4.07	8.69
3	0.19	9.80	20.90
10	0.47	23.64	50.43

Table 2: Broadcast Latency for DSVs

Hard-revoked (#)	Published nodes (#)	Size of DSV list (MB)	Latency (min.)
1000	17,156	0.53	1.13
10,000	138,202	4.25	9.07
100,000	1,049,991	32.32	68.94

- 1,048,900 bits/group $\times 334$ groups ≈ 42 megabytes

Depending on the number of vehicles that need to be revoked, different amounts of compression can be achieved. For example, if 3% of vehicles are revoked, the probability of a bit being 1 is $p = .03$, resulting in an average entropy per bit of

$$\left(p \cdot \log_2\left(\frac{1}{p}\right) + (1-p) \cdot \log_2\left(\frac{1}{1-p}\right)\right) = 0.19, \tag{3}$$

and compression factor of about 0.23 (assuming 20% loss in compression efficiency), i.e. about 10 megabytes. Table 1 provides broadcast latency for SRL with a 64 kbps channel for 3 different scenarios of soft-revoked vehicles. Even in the extreme case of 10% vehicles (i.e., 35 million vehicles) with compromised software, it will only take about 50 minutes to download the entire SRL.

4.5.2 DSVs

It consists of the binary tree encoding followed by the list of published node values. As explained in Section 4.1.3, for a system with a total of n users out of which r users are hard-revoked, number of published binary tree nodes for $1 \leq r \leq n/2$ is about $\left(r \cdot \log_2\left(\frac{n}{r}\right)\right)$ (cf. Theorem 1 of [2]).

Table 2 shows the efficiency of the encoding algorithm presented in Section 4.4. Our binary tree has 2^{40} leaf-nodes. For each of the three revocation scenarios, we ran several iterations for each row in the table, each time randomly picking revoked nodes from a set of 350 million nodes (corresponding to all the vehicles in the system at any point of time). Reported figures are averages over all iterations.

For 10,000 hard-revoked vehicles, which is currently set as the upper limit on the CRL size in the CAMP design, it takes less than 10 minutes to download all the DSVs. On the other hand, in the extreme case of 100,000 hard-revoked vehicles, it takes a little over an hour to download all the DSVs.

To further illustrate the efficiency of our encoding algorithm, let's look at the scenario with 1000 hard-revoked vehicles in a bit more detail. 17,156 node values equal 536 kilobytes in size, as each node is 32 bytes, compared with 540 kilobytes for full packet (including the encoding), i.e. encoding takes less than 1% of the packet. On the other hand, an encoding produced by the basic encoding scheme (that uniquely identifies each node in the tree), would take more than 13% ($< 5/(32+5)$) of the packet.

5 Discussion

In this section we discuss issues related to BCAM:

1. security primitives selected in BCAM in Section 5.1,
2. cost implications of BCAM in Section 5.2,
3. potential risks due to BCAM concept in Section 5.3,
4. operations in Section 5.4.

5.1 Security Primitives

Hash functions like SHA-256 and binary trees built using them are standard cryptographic primitives that have been used to construct numerous secure protocols in the past. In this section, we briefly argue that the design underlying BCAM provisioning is secure assuming that SHA-256 is a Pseudo Random Function (PRF), which is widely believed to be true in the cryptography and security community.

BCAM batches are encrypted using keys that are calculated using a key-derivation function based on HMAC-SHA256, which has been shown to be a PRF assuming the underlying compression function SHA-256 is a PRF [3]. So, if the device picks its key at random, no one other than the CAM and the device itself can derive the BCAM batch encryption key.

Next we argue that a *hard-revoked* device can't use broadcasted DSVs to derive its BCAM batch encryption key. In the binary hash tree that is used in DSV computation, any node in the tree (including leaf nodes) can be computed using any node above on the path of that node to the root node, e.g. parent node, grandparent node, and so on. Moreover, these (nodes on the path to the root node) are the *only* nodes that can be used to compute any given node. When a device is hard-revoked, the leaf node corresponding to that device, and all the nodes on the path from the leaf node to the root node, are excluded from the broadcasted DSVs. So, the collision-resistance (as PRF implies collision-resistance [5]) of hash function like SHA-256 ensures that none of the revoked nodes (and the nodes on the path of the revoked nodes to the root node) can be derived from broadcasted DSVs. Thus, a hard-revoked device won't be able to derive its BCAM batch encryption keys.

5.2 Cost Considerations

We compare this proposal with the current CAMP design for costs under two categories: (1) hardware, and (2) connectivity.

5.2.1 Hardware

Given that most vehicles have satellite radios by default in the US, radio cost is not considered here. The initial storage required for certificates in BCAM is about 8 times bigger than that of CAMP, but we argue that this won't increase the overall cost. As the certificates are stored encrypted in standard, non-secure storage, one could easily leverage spare storage from other in-vehicle system, e.g. satellite module, infotainment system, etc.

5.2.2 Connectivity

Unlike CAMP, in the BCAM approach vehicles don't need connectivity for downloading certificates. Other than that, vehicles can use existing broadcast channels (such as the ones used for downloading CRLs in CAMP) for downloading SRL and DSV list.

5.3 Potential Risks

5.3.1 System Agility

BCAM requires vehicles to be provisioned with their lifetimes worth of certificates. While this alleviates the need for a bidirectional connectivity for certificate downloads, it does mean that a vehicle is locked for the rest of its life into the provisioning decisions (e.g. cryptographic algorithms to use, number of certificates per week, root CA certificate) that were made at provisioning time.

5.3.2 Incorrect Encryption of BCAM Batches

A malicious or malfunctioning RA that gives wrong batches of certificates to CAM for encryption, or that gives the wrong encrypted BCAM batches to the vehicles, will go undetected until later when the DSV list for that batch will be broadcasted. And, even when the DSV list will be broadcasted, the vehicles will have no way of knowing why it can't decrypt the BCAM batches, i.e. whether it has been revoked or it received the wrong (or incorrectly encrypted) batch. Note however that these are also known limitations of the current SCMS design.

5.3.3 Loss of Binary Tree Seed Values by CAM

If a CAM suffers a catastrophic event such that it loses access to its storage (which may include binary tree seed values), all the corresponding vehicles will be locked out and will need to be re-initialized. There is no straightforward mechanism for disaster recovery in this scenario. There are, however, options for disaster management: for example, the RA can retain the original encrypted certificates it received from the PCA and BCAM-enabled devices can fall back to the current CAMP model of opportunistically "topping up" when they get connectivity.

5.3.4 CAM Compromise

If binary tree nodes (especially the root node) are compromised, a vehicle can be given its future DSVs and thereby avoid hard revocation. Therefore, we recommend that if any soft-revoked device managed by that CAM is seen to misbehave, it is added to the CRL and kept on the CRL for the rest of its lifetime. It is clear from this that the physical security of the binary tree nodes is central to the security of the system as a whole. CAM can be made robust against security compromises using standard techniques of secret sharing [21].

5.3.5 Device Key Compromise

If the device key is compromised, the attacker will be able to derive the BCAM batch keys from the DSVs broadcasted by the CAB. This is, however, not a major concern as the certificates returned by the CA are encrypted for the device. Note that if such a compromise is detected, the device should be hard-revoked and re-initialized.

5.4 Operations

5.4.1 Signature on DSV list

We recommend that the DSV list be signed by the CAM that generated it, so that the receiving devices can be certain that they downloaded the right DSV list in its entirety. Note however that this has very little significance in terms of security as an attacker can't forge a DSV, and an incorrect DSV will eventually be detected by the vehicle when it will fail to derive the correct BCAM batch key.

5.4.2 Certificate Lifetime of CA

As the CA needs to issue certificates valid up to 25 years into the future, it needs a certificate with validity of 25 years and more. This has a cascading effect on the certificate lifetimes of CAs up in the hierarchy.

5.4.3 Enforcing Latest SRL in BCAM Batch Key Derivation

The HSM has to use the latest SRL while performing the BCAM batch key derivation, otherwise a software-compromised and soft-revoked vehicle may trick its HSM by passing it an old SRL that doesn't list it as soft-revoked, thereby retrieving the BCAM batch key for decrypting the encrypted BCAM batch. This is done by ensuring that the HSM uses the BCAM period from the SRL in key derivation, so if a malicious vehicle software were to provide an old SRL to the HSM, the resulting BCAM batch key would not be correct and hence the decryption of BCAM batch of certificates would fail.

5.4.4 Multiple CAs

There is a known limitation in the CAMP's current design, in that the first batch of certificates has limited privacy against the CA, as the first batch needs to be generated in a short amount of time and hence the RA doesn't have enough number of requests to shuffle. As vehicles are provisioned with 25 years' worth of certificates in BCAM, they will have limited privacy against CAs for their entire lifetime. This can be alleviated by employing multiple CAs. For maximum privacy, the number of CAs should be equal to the number of concurrently valid certificates (currently set to 20), so that for any vehicle and validity period, a CA would be generating exactly 1 certificate, hence eliminating any possibility of CAs linking two concurrently valid certificates to a single vehicle.

5.4.5 Fail-over

If the BCAM distribution system goes down, legitimate vehicles could be locked out of their certificates because they can't obtain the DSVs. To mitigate this, we propose that the RA store all the certificates issued by the CA (i.e., before BCAM batch encryption), and when the CAM is unavailable, the vehicle download them from the RA similar to the current CAMP approach.

5.4.6 Misbehavior Reporting

Because BCAM does not offer an upload channel, it does not address how misbehavior reporting should occur. As such, it leaves open the question of how and to which vehicles connectivity is provided for misbehavior reporting.

6 Conclusion

Connected vehicles need certificates to participate in V2V communications. To dismiss messages coming from revoked entities, vehicles also need to download a certificate revocation list (CRL) and search through it. The 2-way communication and CRL management (i.e., distribution, processing, etc.) contribute significantly to the overall cost of the system. Therefore, in this paper, we presented a Binary hash tree based Certificate Access Management (BCAM) which has the following advantages over the current CAMP design:

1. **Bidirectional connectivity**: Vehicles do not need bidirectional connectivity for downloading certificates from the SCMS as they are provisioned with all their certificates at the beginning.
2. **Revocation enforced at sender**: By putting a vehicle on the SRL and/or excluding the vehicle from the broadcasted DSVs, a revoked vehicle is prevented from decrypting its certificates. This is in contrast with the current CAMP design where the burden of checking revocation (via CRL) is on the receiving vehicles.
   (a) Vehicles can be removed from the system with surety rather than potentially being trusted by receivers that don't have up-to-date CRL.
   (b) Receivers do not need to store revocation information about revoked vehicles.
   (c) BCAM scales naturally, enabling it to handle revocation rates orders of magnitude higher than the current system.
3. **Unrevoke**: If vehicles have been revoked by error (or, if the vehicles' compromise/malfunction have been properly addressed), they can be easily reinstated by simply removing them from the SRL and/or sending their DSVs again. Note that unrevoking a vehicle that has previously appeared on the CRL is not possible (or, perhaps possible but not implemented) in the current CAMP design.

However, as discussed in this paper, the BCAM approach comes with its set of issues such as weaker system agility and longer (than currently desired) validity periods for CA certificates. Nevertheless, we believe that this approach offers the potential for cost reductions and also offers significant improvements in security management and scalability. More importantly, this approach helps alleviate the concerns around the need for bidirectional connectivity in all vehicles, thereby improving the success probability of NHTSA's proposed regulation on life-saving V2V technologies.

References

[1] National Highway Traffic Safety Administration. Federal motor vehicle safety standards; V2V communication. Technical Report 8, 2017.

[2] William Aiello, Sachin Lodha, and Rafail Ostrovsky. Fast digital identity revocation (extended abstract). In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, pages 137–152, 1998.

[3] Mihir Bellare. New proofs for NMAC and HMAC: security without collision resistance. *J. Cryptology*, 28(4):844–878, 2015.

[4] Daniel R. L. Brown, Robert P. Gallant, and Scott A. Vanstone. Provably secure implicit certificate schemes. In *Financial Cryptography, 5th International Conference, FC 2001, Grand Cayman, British West Indies, February 19-22, 2002, Proceedings*, pages 147–156, 2001.

[5] Ivan Damgård. A design principle for hash functions. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 416–427, 1989.

[6] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In *Pairing-Based Cryptography - Pairing 2007, First International Conference, Tokyo, Japan, July 2-4, 2007, Proceedings*, pages 39–59, 2007.

[7] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.

[8] Amos Fiat and Moni Naor. Broadcast encryption. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, pages 480–491, 1993.

[9] Internet Engineering Task Force. Rfc 4868: Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec. Standard, 2007.

[10] Carlos Gañán, Jose L. Muñoz, Oscar Esparza, Jonathan Loo, Jorge Mata-Díaz, and Juanjo Alins. BECSI: bandwidth efficient certificate status information distribution mechanism for vanets. *Mobile Information Systems*, 9(4):347–370, 2013.

[11] Carlos Gañán, Jose L. Muñoz, Oscar Esparza, Jorge Mata-Díaz, and Juanjo Alins. EPA: an efficient and privacy-aware revocation mechanism for vehicular ad hoc networks. *Pervasive and Mobile Computing*, 21:75–91, 2015.

[12] IEEE. IEEE Std 1609.2-2016 - IEEE Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages. Standard, 2016.

[13] Don Johnson, Alfred Menezes, and Scott A. Vanstone. The elliptic curve digital signature algorithm (ECDSA). *Int. J. Inf. Sec.*, 1(1):36–63, 2001.

[14] Crash Avoidance Metrics Partners LLC. EE Requirements and Specifications Supporting SCMS Software Release 1.1. Technical report, 2016.

[15] Crash Avoidance Metrics Partners LLC. SCMS Proof-of-Concept Interfaces. Technical report, 2016.

[16] Michael Luby. LT codes. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, page 271, 2002.

[17] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, pages 369–378, 1987.

[18] Richard A. Michalski and Ashok Vadekar. *Opportunities for Enhancing the Robustness and Functionality of the Dedicated Short Range Communications (DSRC) Infrastructure Through the Use of Satellite DARS to Improve Vehicle Safety in the 21st Century.* American Institute of Aeronautics and Astronautics, 2016.

[19] National Institute of Standards and Technology. Secure hash standard (SHS). Standard, 2015.

[20] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1):96–99, 1983.

[21] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[22] Mohammad Amin Shokrollahi and Michael Luby. Raptor codes. *Foundations and Trends in Communications and Information Theory*, 6(3-4):213–322, 2009.

[23] William Whyte, André Weimerskirch, Virendra Kumar, and Thorsten Hehn. A security credential management system for V2V communications. In *2013 IEEE Vehicular Networking Conference, Boston, MA, USA, December 16-18, 2013*, pages 1–8, 2013.