Journal of Network and Computer Applications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

COACH: COllaborative certificate stAtus CHecking mechanism for VANETs

Carlos Gañán*, Jose L. Muñoz, Oscar Esparza, Jorge Mata-Díaz, Juan Hernández-Serrano, Juanjo Alins

Universitat Politècnica de Catalunya (UPC), Spain

ARTICLE INFO

Article history:
Received 14 July 2011
Received in revised form
27 December 2011
Accepted 20 February 2012

Keywords:
VANET
Authentication
Certificate validation
PKI
CRL
Merkle hash tree

ABSTRACT

Vehicular Ad Hoc Networks (VANETs) require mechanisms to authenticate messages, identify valid vehicles, and remove misbehaving vehicles. A public key infrastructure (PKI) can be used to provide these functionalities using digital certificates. However, if a vehicle is no longer trusted, its certificates have to be revoked and this status information has to be made available to other vehicles as soon as possible. In this paper, we propose a collaborative certificate status checking mechanism called COACH to efficiently distribute certificate revocation information in VANETs. In COACH, we embed a hash tree in each standard Certificate Revocation List (CRL). This dual structure is called *extended-CRL*. A node possessing an *extended-CRL* can respond to certificate status requests without having to send the complete CRL. Instead, the node can send a short response (less than 1 kB) that fits in a single UDP message. Obviously, the substructures included in the short responses are authenticated. This means that any node possessing an *extended-CRL* can produce short responses that can be authenticated (including Road Side Units or intermediate vehicles). We also propose an extension to the COACH mechanism called EvCOACH that is more efficient than COACH in scenarios with relatively low revocation rates per CRL validity period. To build EvCOACH, we embed an additional hash chain in the *extended-CRL*. Finally, by conducting a detailed performance evaluation, COACH and EvCOACH are proved to be reliable, efficient, and scalable.

1. Introduction

In the last years, wireless communications between vehicles have attracted extensive attention for their promise to contribute to a safer, more efficient, and more comfortable driving experience in the foreseeable future. This type of communications has induced the emergence of Vehicular Ad Hoc Networks (VANETs), which consist of mobile nodes capable of communicating with each other (i.e. Vehicle to Vehicle Communication—V2V communication) and with infrastructure (i.e. Vehicle to Infrastructure Communication—V2I communication). To make these communications feasible, vehicles are equipped with *On-Board Units* (OBUs), and fixed communication units called *Road Side Units* (RSUs) are placed along the road. Finally, multi-hop communication based on IEEE 802.11 is used to facilitate information exchange among network elements that are not in direct communication range (Bera et al., 2006; Jiang and Delgrossi, 2008).

The open-medium nature of these networks makes it necessary to integrate in VANET security mechanisms such as authentication, message integrity, non-repudiation, confidentiality and privacy (Raya and Hubaux, 2005). The solution envisioned to achieve these functionalities is to use digital certificates provided by a centralized certification authority (CA) (Hubaux et al., 2004; Papadimitratos et al., 2007).

In this context, according to the IEEE 1609.2 standard (IEEE, 2006), certificates will be used for digitally signing messages and also for encryption (using the ECIES algorithm). Finally, vehicular networks will rely on a public key infrastructure (PKI) to manage certificates. A critical part of the PKI is how to manage certificate revocation. In general, revocation systems for VANETs can be roughly classified as global or local depending on the extent of the revocation mechanism.

- *Local revocation approaches* enable a group of neighboring vehicles to revoke a nearby misbehaving node. In such approaches, revocation is possible without the intervention of external infrastructure at the expense of trusting other vehicles criteria.
- *Global revocation approaches* are based on the existence of centralized infrastructure such as the PKI, which is in charge of managing revocation.

According to the IEEE 1609.2 standard (IEEE, 2006), vehicular networks will rely on PKI and Certificate Revocation Lists (CRLs)

* Corresponding author. Tel.: +34 934017041.
E-mail addresses: carlos.ganan@entel.upc.edu (C. Gañán), jose.munoz@entel.upc.edu (J.L. Muñoz), oscar.esparza@entel.upc.edu (O. Esparza), jmata@entel.upc.edu (J. Mata-Díaz), jserrano@entel.upc.edu (J. Hernández-Serrano), juanjo@entel.upc.edu (J. Alins).

1084-8045/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2012.02.006

will be used to distribute the status (revoked or valid) of certificates. CRLs are black lists that enumerate revoked certificates along with the date of revocation and, optionally, the reasons for revocation. CRLs in VANET are expected to be quite large because this type of network is expected to have many nodes (vehicles) and also because each vehicle will probably have many temporary certificates (also called pseudonyms) to protect the users' privacy. As a result, a VANET CRL might have a size of hundreds of MB (Nowatkowski et al., 2010; Haas et al., 2011; Wasef and Shen, 2009). The distribution of such a huge structure within a VANET is a challenging issue and it has attracted the attention of many researchers (Papadimitratos et al., 2008a; Laberteaux et al., 2008; Raya and Hubaux, 2005; Haas et al., 2011). A general conclusion about these works is that most of the research efforts have been put into trying to reduce the size of the CRL, either trying to split it or trying to compress it (see Section 2).

In this paper, we take a novel approach because our primary goal is not reducing the CRL size[1] but we aim to design a more efficient way of using the CRL information to distribute certificate status information (CSI) inside the VANET. Our proposal is called COACH (COllaborative certificate stAtus CHecking). COACH is an application-layer mechanism for distributing revocation data. The main idea behind COACH is to embed some little extra information into the CRL such that allows us to create an efficient and secure request/response protocol. For those nodes that just want to obtain status data of some certificates, our protocol replaces downloading a complete CRL. In more detail, we propose a way of efficiently embedding a Merkle hash tree (MHT) (Merkle, 1990) within the structure of the standard CRL to generate a so-called *extended-CRL*. To create the *extended-CRL*, we use an extension, which is a standard way of adding extra information to the CRL. Our extension contains all the necessary information to allow any vehicle or VANET infrastructure element that possesses the *extended-CRL* to build the COACH tree, i.e. a hash tree with the CSI of the CRL. Using this COACH tree, any entity possessing the *extended-CRL* can act as repository and efficiently answer to certificate status checking requests of other vehicles or VANET elements. COACH responses are short since in general, their size is less than 1 kB. This allows a COACH response to perfectly fit within a single UDP message. As we will demonstrate by simulation, this makes the distribution of CSI more efficient than distributing complete CRLs (even though they are compressed), reducing the data that have to be transmitted over the VANET. We must stress that a node possessing an *extended-CRL* can act as COACH repository but that a COACH repository is not a TTP. In other words, COACH is offline, which means that no online trusted entity (like a CA) is needed for authenticating the responses produced by COACH repositories.

Finally, we also propose an enhancement of our basic mechanism called EvCOACH (Evergreen-COACH) to improve the performance of COACH in scenarios with relatively few revocations per CRL validity period. Notice that low revocation rates or small CRL validity periods give rise to such scenarios. In these scenarios, it is plausible to have the same revocation information in several consecutive CRLs. In this case, EvCOACH prevents end-entities from downloading a new CRL whose information is already known. To build EvCOACH, we additionally embed a hash chain in the *extended-CRL*. With this structure, now we can extend the validity of a previous CRL by periodically disclosing successive values of the hash chain. As we will show by simulation, EvCOACH overcomes COACH in terms of bandwidth efficiency in scenarios with relatively few revocations per CRL validity period.

[1] Indeed, our proposal can work together with these other approaches that try to reduce the size of the CRL.

The rest of this paper is organized as follows. In Section 2, we present the background related to our mechanism. In Section 3 we describe in depth COACH. Section 4 depicts EvCOACH, the variant of our proposed mechanism. Section 5 provides a security analysis of the proposals. In Section 6, we evaluate the proposed mechanisms. Finally, Section 7 concludes this paper.

2. Background

In this section, first we start describing the existing global revocation proposals for VANET. Then, we provide a brief overview of Merkle hash trees (MHTs) (Merkle, 1989), which is one of the foundations of the proposed certificate validation mechanism.

2.1. Global VANET revocation mechanisms

Global revocation approaches assume the existence of a Trusted Third Party (TTP), which manages the revocation service. The IEEE P1609.2 standard (IEEE, 2006) proposes an architecture based on CAs. In this architecture, each vehicle possesses several pseudonyms, which are made publicly available by means of short-lived certificates. However, the revocation mechanism for VANET cannot rely uniquely on the use of short-lived certificates (e.g. as proposed in Lu et al., 2008) because compromised or faulty certificates could still cause damage until the end of their lifetimes.

Raya and Hubaux (2005) have proposed the use of short-lived certificates that are preloaded in a tamper-proof device (TPD). The TPD is a trusted component that forms part of the OBU. The TPD stores the valid certificates for a vehicle, signs messages, and performs encryption and decryption functions. Raya et al. introduced two centralized revocation protocols. The first one is based on the revocation of the TPD, which is necessary when all the certificates of a vehicle are to be revoked. This method assumes the presence of the (on-line) infrastructure to send these messages to the trusted component. To ensure that messages from this OBU are not considered valid once the certificates have been revoked, revocation information must also be distributed via CRLs. The second protocol proposed in Raya and Hubaux (2005) is based on the use of compressed CRLs. To compress the CRL, they propose to use Bloom filters. Their method reduces the size of a CRL by using about half the number of bytes to specify the certificate serial number for revocation. Storing CRL information in this manner compresses the size of the CRL considerably since a fixed-length Bloom filter is distributed instead of distributing 8–14 B for every certificate that is revoked.

The distribution of CRLs to all vehicles is not trivial. Some authors Papadimitratos et al. (2007, 2008a) have proposed the use of regional certification authorities instead of using a single central authority. Papadimitratos et al. (2008b) suggest restricting the scope of the CRL within a region. The authors also propose breaking the Certificate Revocation List (CRL) into different pieces and transmitting these pieces using Fountain or Erasure codes. In this way, a vehicle can reconstruct the CRL after receiving a certain number of pieces. Similarly, in Wasef et al. (2010), each CA distributes the CRL to the RSUs in its domain through Ethernet. Then, the RSUs broadcast the new CRL to all the vehicles in that domain. In the case RSUs do not completely cover the domain of a CA, V2V communications are used to distribute the CRL to all the vehicles (Laberteaux et al., 2008). This mechanism is also used in Fan et al. (2008), where it is detailed a public key infrastructure mechanism based on bilinear mapping. Revocation is accomplished through the distribution of CRL that is stored by each user.

Another adaptation of classic public key infrastructure to VANETs is proposed in Armknecht et al. (2007). This architecture

Please cite this article as: Gañán C, et al. COACH: COllaborative certificate stAtus CHecking mechanism for VANETs. Journal of Network and Computer Applications (2012), doi:10.1016/j.jnca.2012.02.006

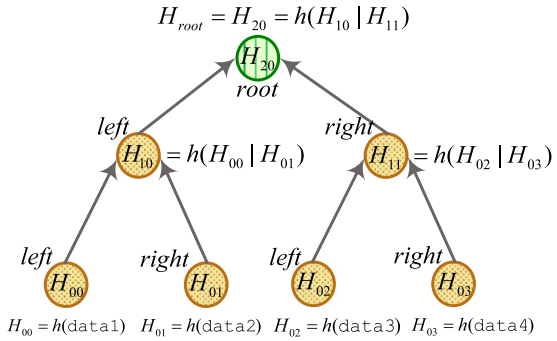

Fig. 1. Sample binary Merkle hash tree.

is based on elliptic curves, where each user gets a master key and a master certificate from the CA. Users can then generate their key pairs or certificates using the masker key, the master certificate and their own secret key. In this mechanism the revocation is also centralized. Whenever a key has to be revoked, the CA publishes some data depending on which the nodes have to update their keys. Lin et al. (2008) present another centralized revocation mechanism, which minimizes the storage at the CA for later liability establishment, but the revocation is aided by the RSU.

Finally, some proposals in the literature divert from the IEEE P1609.2 standard and use the Online Status Checking Protocol (OCSP) (Myers et al., 1999). OCSP is a request/response protocol between clients and responders. An OCSP responder is a trusted intermediate authority for revocation data distribution. Requests may or may not be signed by the client but all the responses must be signed so that clients can ensure that they are communicating with an authorized OCSP responder. In VANET, there is a proposal called ADOPT (Ad Hoc Distributed OCSP for Trust) (Marias et al., 2005) that provides a revocation service based on OCSP in a decentralized manner. ADOPT uses cached OCSP responses that are distributed and stored on intermediate nodes in the VANET.

2.2. The Merkle hash tree

A Merkle hash tree (MHT) (Merkle, 1989) is essentially a tree structure that is built with a One Way Hash Function (OWHF). The leaf nodes contain the hash values of the data of interest (`data1`, `data2`, etc.) and the internal nodes contain the hash values that result from applying the OWHF to the concatenation of the hash values of its children nodes. In this way, a large number of separate data can be tied to a single hash value: the hash at the root node of the tree. MHTs can be used to provide an efficient and highly-scalable way to distribute revocation information, as it is described in Forné et al. (2009) for MANETs (Mobile Ad Hoc Networks). A sample MHT is presented in Fig. 1. This hash tree is binary because each node has at most two children or equivalently, two sibling nodes are combined to form a parent node in the next level. We will denote these siblings as "left" and "right" and a detailed explanation of how to build the hash tree for COACH is given in Section 3.3.

3. COACH: collaborative certificate status checking based on Merkle hash trees

3.1. System requirements

These are the requirements that a status checking mechanism for VANETs must fulfill:

1. *Reliability*: the certificate status checking service must be available at all times, even if RSUs or mobile repositories fail (or are attacked). Reliability is essential to provide security services, and the status checking mechanism must be resilient to attacks. For instance, a denial of service attack should not cause vehicles to consider revoked certificates as valid.
2. *Scalability*: the cost of validating the status of certificates must be low enough. This is due to VANETs may be very large networks, involving from hundreds of CAs to millions of users conducting billions of transactions.
3. *Bandwidth*: the cost in terms of bandwidth consumption should be small. Status checking protocols must minimize the amount of data that has to be exchanged. As a consequence, users will experience shorter validation delays, less burden will be placed on the network infrastructure, and the cost of providing the revocation service will be reduced.
4. *Performance*: the entire system must be efficient, without requiring excessive computational complexity or network overhead for any participant.
5. *Auditability*: while the requirements for the actual revocation decision itself vary widely between implementations, the cryptographic operations should be auditable.
6. *Manageability*: it must be possible to operate the system in a secure manner. Whenever possible, critical keys should be stored in a tamper-proof device where they are less probable to be compromised.
7. *Practicality*: the system must be easy to integrate into existing applications and future infrastructure.
8. *Authentication*: each entity in the network should have an authentic identity. Any received message should first be authenticated before performing further processing.
9. *Unforgetability*: no entity should be capable of generating fake revocation information.
10. *Resistance to replay attacks*: it should be impossible to cheat a vehicle by recording and replaying an old revocation message, for instance trying to persuade this vehicle that a certain certificate is valid when in fact it has been revoked.

3.2. System architecture

The system architecture is an adaptation of a PKI system to the vehicular environment. We present a hierarchical architecture (see Fig. 2) which consists of three levels: the certification authority (CA) is located at level 1, as it is the top of the system. The Road Side Units (RSUs) are located at level 2. Finally, the on-board units (OBUs) are located at the bottom of the hierarchy.

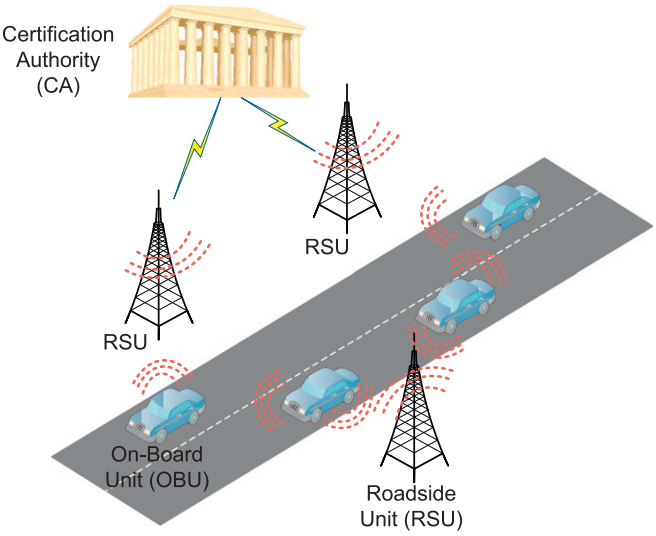

Fig. 2. System architecture.

Please cite this article as: Gañán C, et al. COACH: COllaborative certificate stAtus CHecking mechanism for VANETs. Journal of Network and Computer Applications (2012), doi:10.1016/j.jnca.2012.02.006

The main tasks of each entity are:

1. The CA is responsible for generating the set of certificates that are stored in each OBU. It is also responsible for managing the revocation information and making it accessible to the rest of the entities. By the definition of TTP, the CA should be considered fully trusted by all the network entities, so it should be assumed that it cannot be compromised by any attacker. In fact, in our proposal the CA is the only trusted entity within the network.
2. RSUs are fixed entities that are fully controlled by the CA. They can access the CA anytime because they are located in the infrastructure-side, which does not suffer from disconnections. If the CA considers that an RSU has been compromised, the CA can revoke it.
3. OBUs are in charge of storing all the certificates that a vehicle possesses. An OBU has abundant resources in computation and storage and allows any vehicle to communicate with the infrastructure and with any other vehicle in its neighborhood.

3.3. COACH tree

In this section, we introduce the data structure that COACH uses to handle the revocation service. In this sense, we define the COACH tree as a particular case of Merkle hash tree explained in Section 2. The COACH tree is a binary hash tree where each node represents a revoked certificate.

We denote by $N_{i,j}$ the nodes within the COACH tree, where $i,j \in \{0,1,2,\ldots\}$ represent, respectively, the ith level and the jth node in the ith level. We denote by $H_{i,j}$ the cryptographic (hash) value stored by node $N_{i,j}$.

Nodes at level 0 are called "*leaves*" and they represent the data stored in the tree. In the case of revocation, leaves represent the set Φ of certificates that are revoked at a given instant t,

$$\Phi_t = \{SN_0, SN_1, \ldots, SN_j, \ldots, SN_n\}. \quad (1)$$

If SN_j is the certificate serial number stored by leaf $N_{0,j}$, then $H_{0,j}$ is computed as

$$H_{0,j} = h(SN_j), \quad (2)$$

where $h()$ corresponds to the OWHF function.

Leaves are ordered in the following way: leaves on the left represent smaller serial numbers than leaves on the right. Each node also stores the minimum and maximum serial number of its children. If a leaf has no children, then it uses its own serial number for the maximum and minimum values.

To build the COACH tree, two adjacent nodes at a given level i ($N_{i,j}$, $N_{i,j+1}$) are combined into one node in the upper level, which we denote by $N_{i+1,k}$. Then, $H_{i+1,k}$ is obtained by applying h to the concatenation of the two cryptographic values:

$$H_{i+1,k} = h(H_{i,j} | H_{i,j+1}). \quad (3)$$

At the top level there is only one node called the "root". H_{root} is a digest for all the data stored in the COACH tree. Figure 3 shows a sample COACH tree.

Definition 1. Let the $\mathcal{D}$igest be the concatenation of the certification authority distinguished number DN_{CA}, the root hash H_{root} and the validity period of the CRL. Once created, the $\mathcal{D}$igest is signed by the CA.

$$\mathcal{D}\text{igest} = \{DN_{CA}, H_{root}, ValidityPeriod\}_{SIG_{CA}}. \quad (4)$$

Definition 2. Let the $\mathcal{P}\text{ath}_{SN_j}$ be the set of cryptographic values necessary to compute H_{root} from the leaf SN_j.

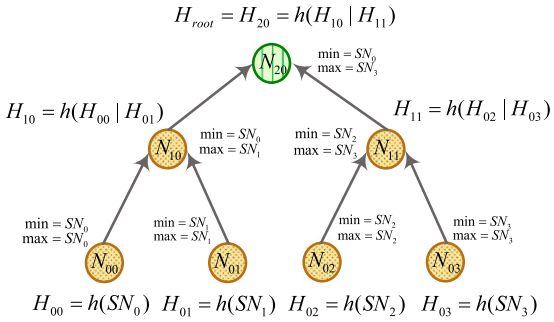

Fig. 3. Sample COACH tree.

Remark 1. Note that the $\mathcal{D}$igest is trusted because it is signed by the CA and it is unique within the tree. Meanwhile, $\mathcal{P}$ath s are different for each leaf.

Claim. An end entity can verify whether $SN_j \in \Phi$ if the MHT provides a response with the proper $\mathcal{P}\text{ath}_{SN_j}$ and the $\mathcal{D}$igest of the MHT.

Example. Let us suppose that a certain user wants to find out whether SN_1 belongs to the sample COACH tree of Fig. 1. Then,

$\mathcal{P}\text{ath}_{SN_1} = \{H_{0,0}, H_{1,1}\}$,
$\mathcal{D}\text{igest} = \{DN_{CA}, H_{2,0}, ValidityPeriod\}_{SIG_{CA}}$.

The response verification consists in checking that $H_{2,0}$ computed from the $\mathcal{P}\text{ath}_{SN_1}$, $h(h(h(SN_1)|H_{0,0})|H_{1,1})$ matches the cryptographic value $H_{2,0} = H_{root}$ included in the $\mathcal{D}$igest

$$H_{root} = H_{2,0} = h(h(h(SN_1)|H_{0,0})|H_{1,1}). \quad (5)$$

Remark 2. Note that the COACH tree can be built by a Trusted Third Party (e.g. a CA) and freely distributed to untrusted repositories. The COACH tree cannot be forged, that is, any change in the tree made by a non-TTP will be detected. This is due to any modification in the COACH tree (for instance, the addition or deletion of a leaf node) causes H_{root} to change. As H_{root} is included in the $\mathcal{D}$igest, which is signed by the CA, this modification will cause the signature to be invalid. To perform a successful attack, the attacker would need to find a pre-image of an OWHF, which is computationally infeasible by definition.

3.4. COACH modus operandi

COACH consists in three stages. During the first stage of *system initialization*, the CA creates the "extended-CRL", that is, a CRL in which a signed extension is appended. This extension will allow the third non-trusted parties to answer status checking requests in an offline way when required. Once this *extended-CRL* has been constructed, it is distributed to the RSUs. In the second stage of *repository creation*, a non-trusted entity (i.e. a RSU or a vehicle) gets the *extended-CRL* and becomes a certificate status checking repository for other VANET entities. Finally, in the third stage of *certificate status checking*, vehicles can use an efficient protocol to obtain the CSI from an available VANET repository.

The main advantage of COACH over CRL is that the entire CRL is not needed for verifying a specific certificate and that a user may hold a succinct proof of the validity of her certificate. With COACH only the repositories must store the whole CRL, while users only need to query for the status of a particular certificate. Thus, the bandwidth that non-repository vehicles need to check the status of a certificate is much lower with COACH than with CRL. On the other hand, COACH allows any vehicle to act as

Please cite this article as: Gañán C, et al. COACH: COllaborative certificate stAtus CHecking mechanism for VANETs. Journal of Network and Computer Applications (2012), doi:10.1016/j.jnca.2012.02.006

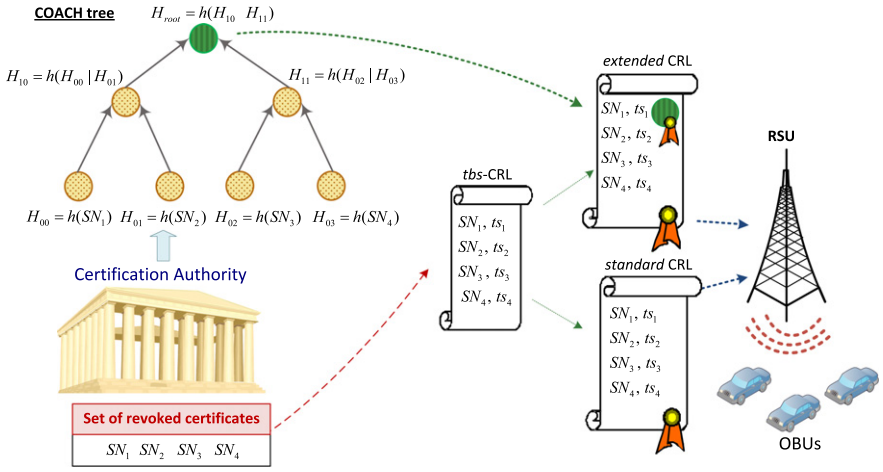

Fig. 4. COACH system initialization example.

repository. This is unfeasible with standard CRLs as the time and the bandwidth necessary to transmit a CRL would not be available with V2V communications. In this way, COACH becomes an offline certificate status validation mechanism as it does not need trusted responders to operate. Henceforward, we give a more detailed description of COACH modus operandi.

3.4.1. System initialization

In this first stage, the CA creates the *extended-CRL* and delivers it to the RSUs. An *extended-CRL* is basically a standard CRL with an appended extension. This extension can be used by non-trusted entities (RSUs and vehicles inside the VANET) to act as repositories and answer to certificate status requests. All the tasks of this system initialization are performed in the CA locally (see Fig. 4).

These are the steps that the CA must carry out:

1. The CA creates a *tbs-CRL*[2] (to be signed CRL), that is, a list that contains the serial numbers of the certificates that have been revoked (along with the date of revocation), the identity of the CA, some time-stamps to establish the validity period, etc.
2. The CA creates the COACH tree, that is, a MHT that is constructed by using the serial numbers within the previous *tbs-CRL* as leaves of the tree. The COACH tree has been designed as a binary tree, and it should be constructed following the methodology explained in Section 3.3. The order of these leaves within the COACH tree is the same than the order of appearance in the tbs-CRL. We assume the *tbs-CRL* to be a sequence of revoked certificate ordered by serial number. Therefore, the leaves of the COACH tree also follow the same order, that is, the bottom left leaf stores the revoked certificate with lowest serial number. Note that if the COACH tree is formed by an odd number n of leaves, there is a leaf $N_{0,n-1}$ that does not have a pair. Then the single node is simply carried forward to the upper level by hashing its $H_{0,n-1}$ value. We proceed in the same way if any ith level is formed by an odd number n of nodes. Once created the MHT, the CA obtains the root hash.
3. The CA calculates the extension, which consists basically of the $\mathcal{D}$igest. Just recall that this $\mathcal{D}$igest was calculated in Eq. (4) as the concatenation of the certification authority distinguished number, the root hash and the validity period of the CSI, and after that signed by the CA. Obviously, the distinguished number and the validity period should be the same than the ones contained in the tbs-CRL. In fact, the COACH tree is just a different way of representing the CSI, but the hash tree will be valid during the same time and will provide the same information than the CRL. Once calculated, this $\mathcal{D}$igest is appended to the tbs-CRL, generating the *tbs-extended-CRL*.
4. The CA signs the tbs-extended-CRL, generating the *extended-CRL*. Notice that this second overall signature not only authenticates all the CSI, but also binds this CSI to the $\mathcal{D}$igest. The *extended-CRL* is only slightly larger than the standard CRL, as we will show later in Section 6.
5. Finally, the CA distributes copies of the *extended-CRL* to the designated RSUs, which will act as the typical PKI repositories, in the same manner as they would do with a standard CRL.

After this first stage of system initialization, the RSUs have a copy of the *extended-CRL*, which contains exactly the same CSI than a standard CRL and it is valid for the same time. The advantage of an *extended-CRL* is that any non-trusted entity in possession of it can generate again the COACH tree locally, and obtain the root hash. As the *extended-CRL* also includes the $\mathcal{D}$igest, which is signed by the CA, this entity has an authenticated version of the COACH tree and can answer to CSI requests in an offline way.

3.4.2. Repositories creation

In this stage, RSUs become new repositories of the VANET. Vehicles can also become mobile repositories allowing to distribute the extended-CRL information in areas with poor coverage. To become a repository an entity must follow the next steps:

1. The entity obtains the *extended-CRL* either from the CA or from another entity that has an up-to-date copy of the *extended-CRL* in its cache. Notice that the CA uses a secure wireline to communicate with the RSUs, while the RSUs use a wireless link to communicate with the vehicles.
2. Once the *extended-CRL* has been downloaded, the entity verifies that the signature of the *extended-CRL* is valid and corresponds to the CA. If so, the entity generates locally the COACH tree using the serial numbers within the *extended-CRL* and following the same algorithm than the CA (as explained in Section 3.3). The root hash of the tree created from the *extended-CRL* entries must match the signed root value contained in the $\mathcal{D}$igest.
3. At this moment, the entity can respond to any status checking request from any vehicle until the *extended-CRL* expires.

[2] The CA can generate the standard CRL by simply signing this tbs-CRL.

3.4.3. Certificate status checking

After the second stage, RSUs and some vehicles will be able to act as repositories. The last stage of the mechanism consists in providing the certificate status information to any vehicle that needs to validate the status of a certificate. Firstly, a vehicle that needs to check the status of a certificate must locate a valid repository. To do so, the vehicle uses a service discovery protocol (SDP) to find a RSU or a vehicle that is acting as repository. A secure service discovery and communication protocol is mandatory in order to prevent from many attacks and malicious processes in VANETs. We assume that the VANET is using an efficient service discovery protocol that guarantees a secure discovery and communication in the vehicular system while maintaining the network scalability and a low communication delay. There have recently appeared some new protocols for ad hoc environments that intend to provide these features (Moschetta et al., 2010; Kim et al., 2011; Abrougui et al., 2010; Abrougui and Boukerche, 2011). For instance, authors in Abrougui and Boukerche (2011) design an advertisement and discovery mechanism for VANETs which permits vehicles to discover nearby RSUs and their services securely and preserving their privacy. Any of these mechanism could work with COACH to facilitate the discovery of the repositories.

Once the repository has been located, the vehicles start the status checking protocol. The protocol for status information exchange is based on the hash tree structure and it allows checking the integrity of a single *extended-CRL* entry with only some hash material plus the $\mathcal{D}$igest (included in the extension). On the one hand, this is much more efficient than broadcasting the entire *extended-CRL*. On the other hand, the mechanism is fully offline (the only trusted authority is the CA), which is a very good feature because sometimes it may be impossible for vehicles to reach the CA due to lack of coverage.

Hence, a vehicle that needs to check the status of a certificate must follow the next steps:

1. The vehicle uses a service discovery protocol to find either a RSU or a mobile repository inside its coverage range for status checking.
2. The vehicle sends the serial number of the certificate that is going to be verified to the repository. The repository searches the target certificate in the hash tree. In the case the certificate is found, the repository sends the $\mathcal{P}$ath, i.e. the hash values of the nodes of the tree which are needed to calculate the signed root.
3. The vehicle verifies that the H_{root} calculated from the $\mathcal{P}$ath matches the H_{root} contained in the $\mathcal{D}$igest.

Notice that as the H_{root} is signed by the CA, it is just as impractical to create falsified values of the $\mathcal{P}$ath as it is to break a strong hash function. In case the certificate is not revoked, the repository sends the adjacent leaves to the requested certificate. To this respect, the repository has to prove that a certain certificate (SN_{target}) does not belong to the set of revoked certificates (Φ). To prove that $SN_{target} \notin \Phi$, as the leaves are ordered, it is enough to demonstrate the existence of two leaves, a minor adjacent (SN_{minor}) and a major adjacent (SN_{major}) that fulfill:

1. $SN_{major} \in \Phi$.
2. $SN_{minor} \in \Phi$.
3. $SN_{minor} < SN_{target} < SN_{major}$.
4. SN_{minor} and SN_{major} are adjacent nodes.

In any case, the data that the repository needs to send to a node to perform the status checking can be placed in a single UDP datagram using 802.11p link-layer.

It is worth noting that under low-equipped vehicle scenarios, communication disruptions may occur frequently due to high mobility, network congestion, or potential attacks. In such scenarios, requesting nodes that do not get an answer from a local repository have to assume that either its request/respond has not arrived (e.g. due to possible interferences), the repository has been compromised or a malicious node is acting as prankster. Note that in any case, the requesting node assumes the absence of the repository because the physical layer in VANETS (based on the Dedicated Short Range Communication (DSRC) protocol Jiang and Delgrossi, 2008) defines a control channel where every node broadcasts a beacon that provides trajectory and other information about the vehicle. Thus, any node is aware of its (legitimate) neighbors at any instant and which are acting as mobile repositories. At this point, reaching a local repository becomes a routing problem which has been well studied in the literature (e.g. Jain et al., 2004; Chen et al., 2010).

4. Evergreen COACH (EvCOACH)

In this section, we present a variant of the COACH mechanism specially designed to enhance performance when the revocation rate is low. As explained in the previous section, COACH computes a new hash tree each time the *extended-CRL* expires. However, it could be the case where there have been no revocations during the lifetime of the previous *extended-CRL*. That is to say, the list of revoked certificates has not changed during successive update periods, and the only parameter that has changed is the validity period of the *extended-CRL*. Thus, the set of revoked certificates at $t_0 = thisUpdate$ is the same as the set of revoked certificates at $t_1 = nextUpdate$, i.e. $\Phi_{t_0} = \Phi_{t_1}$. In this context, the *extended-CRL* has a lifetime equal to $nextUpdate - thisUpdate$.

In this case, vehicles that already have downloaded the whole *extended-CRL* do not obtain new information by downloading a new CRL that has no new revoked certificates. Thus, it would be highly inefficient to publicize a whole new *extended-CRL* when only the timestamps are different from the previous *extended-CRL*. For this reason, we propose a new mechanism that reutilizes the COACH tree calculated previously, minimizing the amount of information that has to be transmitted in such situation. For this purpose, we have designed the EvCOACH tree, which is a COACH tree that is constructed by adding a new branch. This new branch is calculated by hashing a new secret nonce generated by the CA (see Fig. 5). In this context, the new EvCOACH tree can be considered perennial as it can live for more than one *extended-CRL* lifetime. This "perennial" property means that the validity of the hash tree can be reassured as many times as the length of the hash chain.

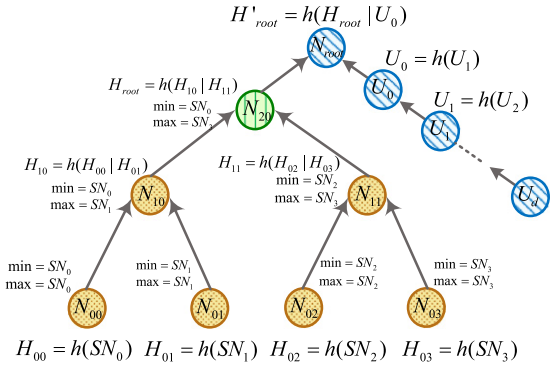

Fig. 5. Example of EvCOACH tree.

Before describing EvCOACH, we introduce some new definitions.

Definition 3. Let *primaryUpdateValue* (U_d) be a secret nonce that the CA generates.

Definition 4. Let *maximumUpdateValue* (U_0) be the parameter that is concatenated with the old hash root to compute the new root of the EvCOACH tree.

Definition 5. Let *currentUpdateIndex* (i) be the number of time-periods (Δt) elapsed since the last *extended-CRL* publication.

Definition 6. Let *maximumUpdateIndex* (d) be the maximum number of time-periods that the *extended-CRL* can be revalidated.

Next equation shows the relations among these definitions. Note that each value U_i can be calculated applying a hash function h to the previous value, and the first value of the hash chain is the secret nonce U_d.

$$U_d \xrightarrow{h} U_{d-1} \xrightarrow{h} \cdots \xrightarrow{h} U_i \xrightarrow{h} U_2 \xrightarrow{h} U_1 \xrightarrow{h} U_0.$$

Definition 7. Let *validityInterval* (Δt) be the additional amount of time during which an invariant EvCOACH tree is still valid.

The aim of this new mechanism is to minimize the amount of information that has to be sent when there are no new revoked certificates with respect to the previous *extended-CRL* publication. For that purpose, the mechanism revalidates the previous *extended-CRL* during a certain amount of time (Δt) without having to recalculate the EvCOACH tree again. Note that the duration of Δt is critical to the performance of the mechanism. The over-estimation of Δt could allow to operate with certificates even though they have been revoked. On the contrary, the under-estimation of Δt could lead to over-issuing CRLs. We give some hints on how to estimate Δt accurately in Section 4.4. We now give details for the operations of the three parties in the system (CA, RSUs and vehicles) when using EvCOACH.

Note that EvCOACH is just an extension of COACH that allows reassuring the validity of the MHT when there are no new revocations. This is achieved at expenses of an additional communication overhead and computational cost. However, the benefits of not updating the *extended-CRL* overcome these costs.

4.1. CA operations

- *System initialization*:
  1. *Creation of the EvCOACH tree*: the CA generates one part of the EvCOACH tree by using the set of initially revoked certificates, as explained in Section 3.3. Notice that at the top level of the COACH tree, there is only one node whose value is H_{root}. In fact, this part of the EvCOACH tree has been constructed in the same way than a COACH tree (see Section 3.4.1). The CA also has to generate the other part of the EvCOACH tree, the new branch. To do so, the CA generates a nonce (U_d) that must be kept secret during the whole validity period of the hash tree. Then, it calculates U_0 applying d times the hash function to U_d:

     $$U_0 = h^d(U_d).$$

     Then, the EvCOACH tree is finally the union of both parts, the COACH tree and the new branch, as it is shown in Fig. 5. The new EvCOACH root hash can be calculated as

     $$H'_{root} = h(H_{root} | U_0).$$

  2. *Creation of the* $\mathcal{D}$*igest*: the way in which EvCOACH calculates the $\mathcal{D}$igest is slightly different from COACH, because it includes the first node of the new branch. First, the CA estimates the amount of time (Δt) during which the CRL is not expected to change. That is to say, it estimates the amount of time Δt during which no new certificates are expected to be revoked. Finally, the EvCOACH $\mathcal{D}$igest is calculated as

     $$\mathcal{D}\text{igest} = \{DN_{CA}, H'_{root}, U_0, \Delta t\}_{SIG_{CA}}.$$

     Again, this $\mathcal{D}$igest is included within the *extended-CRL*, as it was in COACH.
  3. *Distribution of the revocation material*: finally, the CA sends to the RSUs the *extended-CRL*, which contains the (sorted) list of revoked certificates serial numbers along with the EvCOACH $\mathcal{D}$igest.
- *Revalidation of the revocation material*: Once the *extended-CRL* expires, the CA must check whether it can re-use the revocation material already distributed. To that end, the CA has to check if there are new revoked certificates. If there are new certificates the CA must initialize the system again, recalculating the EvCOACH tree (using a new $U_{d'}$) and the $\mathcal{D}$igest, and distributing all the revocation material. On the contrary, if the list of revoked certificates is the same, so that:

  $$\Phi_{t_0} = \Phi_{t_1} = \Phi_{t_1 + i\Delta t}, \quad i \geq 0,$$

  then, the CA can revalidate the revocation material performing the following steps (just remember that $t_0 = thisUpdate$ and $t_1 = nextUpdate$):

  1. *Calculation of the new* U_i: the CA calculates the new value U_i where i is the number of time-periods (Δt) elapsed since the first CRL publication:

     $$U_i = h^{d-i}(U_d).$$

     Note that at a given instant t_i the CA is the only one capable of calculating the U_i. However, any node that receives a nonce U_i can check its authenticity by hashing i times U_0.
  2. *Distribution of the freshest* U_i: the corresponding U_i is distributed to the RSUs and mobile repositories. Notice that the size of the U_i is lower than the size of the $\mathcal{D}$igest distributed in COACH and much lower than the CRL. In this sense, the CA revalidates the information of the EvCOACH tree just by issuing a nonce.

4.2. RSU operations

- *Retrieving the revocation material*: at CRL update instants, RSUs obtain the *extended-CRL* from the CA through a secure wireline. If instead of an update there is a revalidation of the CRL, the RSUs just download the corresponding U_i.
- *Responding to vehicles requests*: in order to be able to respond to the users' certificate status queries, RSUs must manage the EvCOACH tree. This hash tree can be created directly from the *extended-CRL*, or revalidated by checking the validity of the corresponding U_i. When a user's request arrives at the RSU, it has to process whether it is a request for the whole CRL, or just a certificate status checking. In the former case, the RSU has to forward the *extended-CRL* that it has previously downloaded from the CA. In the latter case, it just has to calculate the $\mathcal{P}$ath of the certificate to prove that a certificate is revoked, or calculate SN_{minor} and SN_{major} to prove the contrary.

4.3. Vehicle operations

Regarding the operations that a vehicle can perform, we can distinguish two types of vehicles, those who want to contribute to

Please cite this article as: Gañán C, et al. COACH: COllaborative certificate stAtus CHecking mechanism for VANETs. Journal of Network and Computer Applications (2012), doi:10.1016/j.jnca.2012.02.006

the revocation mechanism and become repositories, and those who just want to check the status of the certificates. Repository vehicles can perform the same actions as normal vehicles but, in addition, they can also perform RSU operations. It must be noted that when repository vehicles perform RSU operations they do not communicate with the CA but with the RSUs in range.

- *Checking certificate status*: when a vehicle needs to communicate with another entity, firstly it must check the validity of the entity's certificate. For that purpose, if there is a reachable RSU, a non-repository vehicle queries it for the $\mathcal{D}$igest or the corresponding U_i and the $\mathcal{P}$ath of the certificate. Then, it checks the validity of the $\mathcal{D}$igest. To do so, they verify that the corresponding U_i is valid as this value can only be generated by the CA due to the properties of a OWHF. Therefore, to check validity of U_i at an instant $t' \in [nextUpdate+(i-1)\Delta t, nextUpdate+i\Delta t]$ a vehicle has to verify that the following equality is satisfied

  $$U_0 = h^i(U_i) \quad \text{with } i \leq d.$$

  Once, it has been checked the validity of the U_i, the vehicle has to check whether the $\mathcal{P}$ath is valid or not. To do so, the vehicle verifies that the H'_{root} calculated from the $\mathcal{P}$ath matches the H'_{root} contained in the $\mathcal{D}$igest.
- *Responding to vehicle requests*: repository vehicles can also respond to other vehicle requests, following the same steps than a RSU.

Note that the main differences between COACH and EvCOACH are: (1) EvCOACH has an additional branch in the MHT that allows the revalidation of the MHT and (2) the $\mathcal{D}$igest in EvCOACH contains one extra parameter to allow the verification of the revalidation. With these slight modifications, EvCOACH highly improves its performance in those VANET where the CRL does not grow between consecutive updates.

4.4. Estimating Δt

In order to estimate the duration of the validity of the certificate status information, the CA needs to balance the liability cost of not releasing a new CRL on time and the costs of releasing CRLs too often. We show a technique to calculate the revalidation interval (Δt) of the EvCOACH tree to minimize that cost.

Authors in Ma et al. (2006) carry out an analytical study dealing with optimization of the release of CRLs. Although the data that they collected to perform the analysis do not belong to a CA deployed in a VANET, we can extrapolate some of their results to our scenario. According to Ma et al. (2006) and Walleck et al. (2008), the probability density function (PDF) of the revocation process fits an exponential distribution

$$f(x) = \frac{1}{\mu} e^{-x/\mu}.$$

The μ parameter represents the mean lifetime of a revoked certificate, and according to those papers it equals 30 days approximately. With that information and the lifetime of the operative certificates in the VANET, the CA can estimate the probability of having a certificate revoked, and thus, estimate Δt.

5. Security analysis

In this section we provide a brief security analysis of COACH and EvCOACH. As a general remark, note that it is relatively straightforward to see that these structures are secure because they are essentially a type of Merkle hash tree, which has been proven to be secure.

In first place, assuming that h is a one-way collision-resistant hash function and in the second place, assuming that the signature scheme (i.e. ECDSA) is secure, neither a proof of revocation nor a proof of validity can be forged. Thus, any integrity violation of a node in a tree can be detected thanks to the properties of the hash function h, which is one-way and collision-resistant. Next, we show that COACH and EvCOACH are secure against fabrication attacks, denial of service attacks and simple replay attacks.

5.1. Fabrication attacks

In COACH, a $\mathcal{P}$ath is legitimate only if it contains the signed proof (i.e. the root of the Merkle hash tree) and it has not expired. The one-way property of the hash function prevents any adversary from fabricating a $\mathcal{P}$ath that yields the correct value for the COACH-tree root. Note that the same applies to EvCOACH as the proof included in the $\mathcal{P}$ath is calculated by hashing a random value (U_i) and the root of the COACH-tree. As U_i is calculated by hashing i times a nonce only known by the CA, no entity can forge this value. Therefore, both mechanisms are resistant against fabrication attacks.

5.2. Replay attacks

COACH and EvCOACH are also safe against replay attacks in which a malicious entity is replying with an old $\mathcal{P}$ath. This is because the $\mathcal{D}$igest includes a validity period, so any entity can check if a given $\mathcal{P}$ath that has associated a specific the $\mathcal{D}$igest is outdated. This validity period cannot be forged or altered thanks to the one-way property of the hash function. Therefore, expired $\mathcal{P}$ath s can be dismissed, effectively avoiding this kind of replay attacks.

5.3. Denial of service attacks

Finally, malicious and selfish behaviors should be considered when providing revocation data in a VANET. For example, a malicious vehicle in a VANET may start flooding the network with requests. As a result, mobile repositories and RSUs receiving these bogus requests will calculate the corresponding $\mathcal{P}$ath to build the responses. This process consumes resources and might potentially deny the service to other vehicles. On the other hand, some VANET nodes may exhibit a selfish behavior. In terms of COACH and EvCOACH, selfish vehicles may, for instance, decide not becoming a repository despite having enough resources for doing so. These behaviors on a large scale VANET can result in network congestion and potential denial of service. To prevent and deal with these attacks, a trust establishment framework could be used to support COACH's operation (e.g. Marias et al., to appear). This type of frameworks, incorporate self-evidences, recommendations, subjective judgment and historical evidences to continuously evaluate the trust level of vehicles, which can benefit COACH (and EvCOACH) in terms of availability and robustness.

6. Performance evaluation

6.1. Analytical evaluation

In this section, we analytically compare the performance of COACH with other mechanisms in terms of overhead and computational cost.

Please cite this article as: Gañán C, et al. COACH: COllaborative certificate stAtus CHecking mechanism for VANETs. Journal of Network and Computer Applications (2012), doi:10.1016/j.jnca.2012.02.006

6.1.1. Communication overhead

Lets start estimating the size of a CRL in a vehicular environment. The size of a CRL is proportional to the number of revoked certificates. Let N_{veh} be the total number of vehicles in the region that the CRL needs to cover, ρ the average percentage of certificates revoked, L_f the lifetime (or validity period) of a certificate, and $\bar{s}$ the mean number of pseudonyms of a vehicle. Additionally, let N_{rev} be the number of non-expired certificates that were revoked, i.e. the number of certificates that the CRL contains. According to Walleck et al. (2008), the probability density function of certificate revocation approximately follows an exponential distribution:

$$f(t;L_f) = L_f \cdot e^{-t \cdot L_f}, \quad \forall t \geq 0.$$

Therefore, if certificate is revoked at instant t of its lifetime, it stays in the CRL for $L_f - t$. Thus, the expected time a revoked certificate stays in the CRL can be estimated as

$$E(L_f - t) = E(L_f) - E(t) = L_f - \frac{1}{L_f} = \frac{L_f^2 - 1}{L_f} \simeq L_f.$$

Then, we can estimate the mean number of revoked certificates in a CRL as

$$\overline{N_{rev}} = N_{veh} \cdot \rho \cdot \bar{s} \cdot L_f.$$

Finally, we estimate the size of a CRL in a VANET. The CRL contains some header information and a CRL entry for each certificate on the list. CRL entries will have varying sizes, but according to 1609.2 standard (IEEE, 2006), 14 B per entry is a realistic figure, i.e. $s_e = 14$ B. The size of the CRL header is negligible compared to the total size of the CRL. According to NIST statistics (Berkovits et al., 1995), 10% of the certificates need to be revoked during a year, i.e. $\rho = 0.1$. Recall that in a VANET, each vehicle owes not only an identity certificate, but also several pseudonyms. The number of pseudonyms may vary depending on the degree of privacy and anonymity that it must be guaranteed. According to Raya, Papadimitratos, and Hubaux in Haas et al. (2009) the OBU must store enough pseudonyms to change pseudonyms about every minutes while driving. This equates to about 43,800 pseudonyms per year for an average of 2 h of driving per day. Haas, Hu, and Laberteaux in Papadimitratos et al. (2008a) recommend changing pseudonyms every 10 min, and driving 15 h per week. This equates to 4660 pseudonyms per year, but they recommend storing five years of pseudonyms for a total of about 25,000 pseudonyms per OBU. Therefore, we set $\bar{s} = 25,000$. Regarding to the certificate lifetime, according to Walleck et al. (2008), it ranges from 26 to 37 days. In this manner, we set the lifetime to 1 month. Therefore, the expected CRL size is

$$\text{CRL}_{size} = \overline{N_{rev}} \cdot s_e = N_{veh} \cdot \rho \cdot \bar{s} \cdot L_f \cdot s_e.$$

Assuming that a regional certification authority could manage around 50,000 vehicles, the expected CRL size is $\text{CRL}_{size} \simeq 145$ MB. On the other hand, the response size of COACH is much smaller than a CRL as it consists only of the $\mathcal{D}$igest and the $\mathcal{P}$ath for a given certificate. Using the SHA-1 algorithm (hash size of 160 bits), and ECDSA-256 the size of the response of COACH for 10,000,000 revoked certificates (including pseudonyms) is of approximately 710 B. In the same way, EvCOACH has the same response length of standard COACH but adding one nonce to the response (typically 15 B).

Table 1 shows the size of the response and the request for the different certificate validation mechanisms. Note that the request size is very similar for all the mechanisms. However, the size of the response varies significantly, e.g. COACH and EvCOACH response sizes are six orders of magnitude smaller than conventional CRL. Figure 6 shows the size of the response for CRL, ADOPT (Marias et al., 2005) and COACH depending on the number of revoked vehicles in the network. While ADOPT response size is constant, the size of the response when using CRL or COACH increments with the number of revoked certificates. As you can observe, the CRL size grows linearly with the number of revoked certificates, while COACH and EvCOACH response sizes describe a logarithmic growth.

Table 1
Comparison of the overhead introduced by COACH and other certificate validation mechanisms.

Mechanism	Request size (B)	Response size
CRL	73	145 MB
COACH	73	710 B
EvCOACH	73	725 B
ADOPT	66	586 B

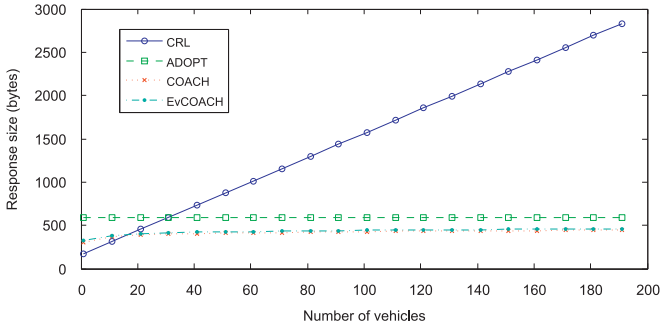

Fig. 6. Response size vs number of vehicles.

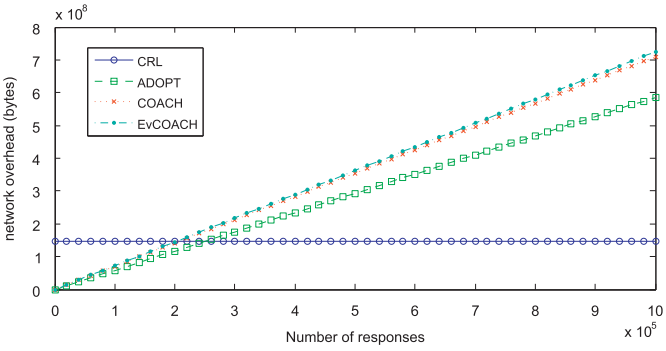

Fig. 7. Network overhead vs number of responses.

Notice that downloading a CRL is more efficient than performing COACH requests/responses only when the vehicle makes more than 200,000 requests (see Fig. 7), which is quite unfeasible in a typical VANET environment where a normal vehicle is not expected to establish contact with such amount of vehicles. Regarding ADOPT, its response size is slightly smaller than in COACH, but it lacks the benefits that COACH provides to operate during disconnections. Essentially, ADOPT is a caching mechanism for OCSP responses. Like COACH responses, OCSP responses are much smaller than VANET CRLs and thus, under a low request rate ADOPT can feasibly provide timely revocation status information without burdening the network. The main problem with ADOPT is related to how to locate cached responses. Vehicles have to broadcast their queries in the hope of finding another vehicle that has previously asked for the desired CSI and that has this cached data. This broadcast incurs a significant overhead in a VANET, where vehicles have to validate several certificates per second. Moreover, even if a vehicle is able to locate an intermediate node capable of serving the desired CSI, this CSI

could be outdated. Therefore, ADOPT cannot assure that a response has been previously cached by a neighbor vehicle in such dynamic scenarios, in other words, it is quite probable that vehicles will end up querying an OCSP responder or the CA for the CSI. In vehicular scenarios the number of cached responses could be huge, and therefore, also a huge storage capacity is required in the vehicle. In addition, ADOPT does not guarantee that a vehicle obtains the status of a given certificate when needed. So, ADOPT has smaller responses, but it does not provide as fresh information as COACH and it forces VANET nodes to store a large amount of CSI data. Finally, ADOPT makes the network more vulnerable than when using COACH because more trusted connections with OCSP responders or the CA are needed.

6.1.2. Computational cost

As an initial remark, we would like to mention that in the following evaluation, we consider the cryptographic delay only due to hashing and point multiplication on the elliptic curve, as they are the most time-consuming operations in the proposed protocol. Let T_{hash} and T_{mul} denote the time required to perform a pairing operation and a point multiplication, respectively. The elliptic curve digital signature algorithm is the digital signature method chosen by the VANET standard IEEE1609.2, where a signature generation takes T_{mul} and a signature verification takes $4T_{mul}$. In COACH, to verify a credential, a verifier must perform a hash operation to compute the current contents of the leaf node corresponding to the target serial number (SN_i). Finally, the verifier needs to perform $\log N$ hash operations to compute the root of the tree using the $\mathcal{P}$ath. Therefore, the total computation overhead when checking the status of a certificate is $T_{hash}(\log N+1)+4T_{mul}$. In Zhang et al. (2008), T_{mul} are found for an MNT curve with embedding degree $k=6$ that is equal to 0.6 ms. In our simulation, we use an Intel Core i7 950 (at 3.07 GHz) which is able to perform 1015952 SHA-1 Hashes per second, i.e. $T_{hash}=0,98$ μs.

On the other hand, EvCOACH introduces an additional delay due to the validation of the nonce transmitted along with the $\mathcal{P}$ath. In this case, the verifier must hash the nonce i times (where i represents the *currentUpdateIndex*) to obtain U_0. Once U_0 is calculated, an additional hash operation is needed to obtain the new root hash (i.e. H'_{root}). Therefore, the total computation overhead due to a certificate status checking in EvCOACH is $T_{hash}(\log N+i+2)+4T_{mul}$.

Table 2 shows the verification and signing delays of verifying k certificates for each revocation system. CRL is the best mechanism in terms of computational cost, as the CA only needs to sign once the CRL and vehicles only have to check this signature to validate the status of any certificate. ADOPT is based on OCSP in which typically each response is digitally signed, which imposes a high computational cost on responders. Furthermore, since the ADOPT responder is a trusted online server, its security requirements are stricter compared with a CRL distribution server. Finally, COACH and EvCOACH have a little more computational cost than CRL but much lower than ADOPT. This is mainly due to fact that the CA only has to sign the H_{root}, and vehicles only need to check this signature to validate each certificate. However, for each certificate this root has to be calculated which induces a verification delay that does not exist with CRL.

Table 2
Computational cost of validating k certificates per revocation mechanism.

Mechanism	Verification delay	Signing delay
CRL	$4T_{mul}$	T_{mul}
COACH	$k(T_{hash}(\log_2 N+1)+4T_{mul})$	T_{mul}
EvCOACH	$k(T_{hash}(\log_2 N+i+2)+4T_{mul})$	T_{mul}
ADOPT	$k(4T_{mul})$	$k(T_{mul})$

6.2. Evaluation setup

In the previous section, we have shown analytically that COACH and EvCOACH mechanisms outperform CRL in terms of bandwidth efficiency and time to get fresh CSI. Moreover, COACH also improves other revocation mechanisms such as ADOPT when analyzing the availability of fresh CSI. In this section, we evaluate the proposed mechanism in a VANET scenario taking into account the specific characteristics of these networks.

A novel emulator tool (Reñé et al., 2010) has been used to carry out the evaluation of COACH. Its accuracy and reliability is shown in Reñé et al. (2011). This emulation platform allows to execute real-time applications over VANET, but for our purpose, we only use the simulation capabilities of this platform. The network emulation that this platform implements is based on the Network Simulator NS-2 (McCanne and Floyd, 2000) with some add-ons to enhance its functionality. To this respect, we use an open source micro-traffic simulator called SUMO (Krajzewicz et al., 2002) to generate a realistic mobility model so that results from the simulation correctly reflect the real-world performance of a VANET (see Fig. 8). Thus, the results obtained with this platform are equivalent to the ones that would result from simulating the same scenario using SUMO and NS-2.

Using this emulation platform for VANET, we have evaluated the performance of our mechanism under three different environments. The reference scenario for all three environments is shown in Fig. 9. This scenario consists of 4 two-lane roads forming a 1000×500 m rectangle. RSUs are placed every 300 m along the highway in order to achieve full coverage.

Table 3 summarizes the values of the configuration parameters used in the reference scenario. Note that we have configured our simulation to use the Nakagami propagation model. We choose this propagation model because empirical research studies have shown

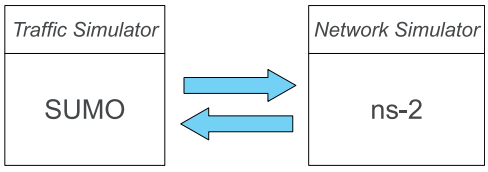

Fig. 8. Simulation architecture.

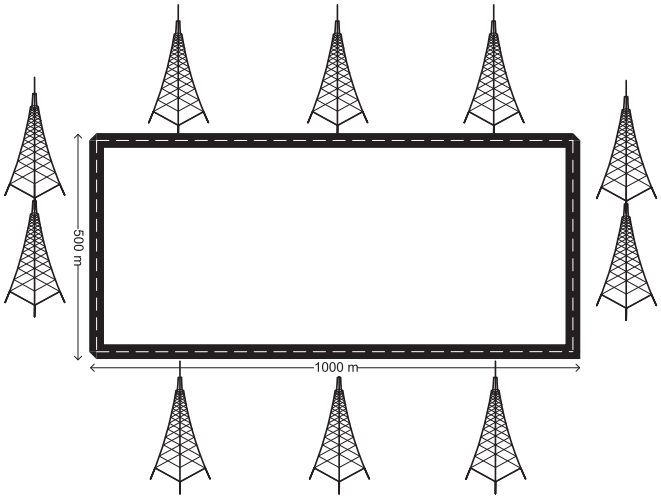

Fig. 9. Reference scenario.

Please cite this article as: Gañán C, et al. COACH: COllaborative certificate stAtus CHecking mechanism for VANETs. Journal of Network and Computer Applications (2012), doi:10.1016/j.jnca.2012.02.006

Table 3
Parameter values for the reference scenario.

Parameter	Value
Area	1000 × 500 m
Number of lanes	2
Number of RSUs	10
RSU Transmission range	300 m
MAC	IEEE 802.11p
Bandwidth of a channel	10 MHz
Propagation model	Nakagami
Transport protocol	UDP

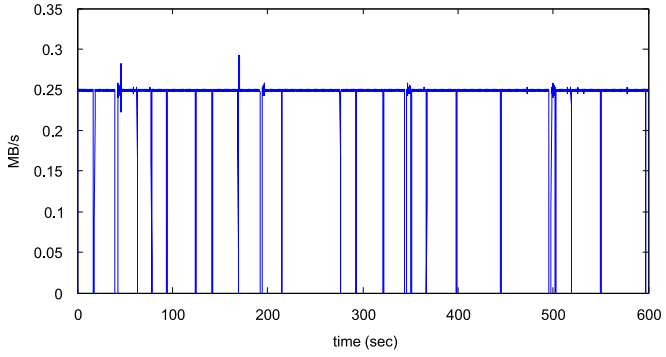

Fig. 10. Throughput for one single car at 72 km/h.

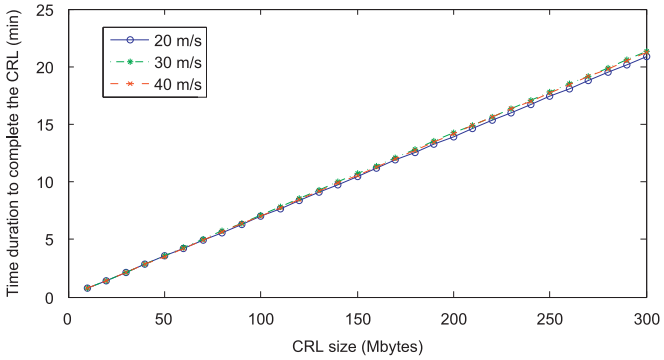

Fig. 11. Time to download a CRL for a single vehicle at different speeds.

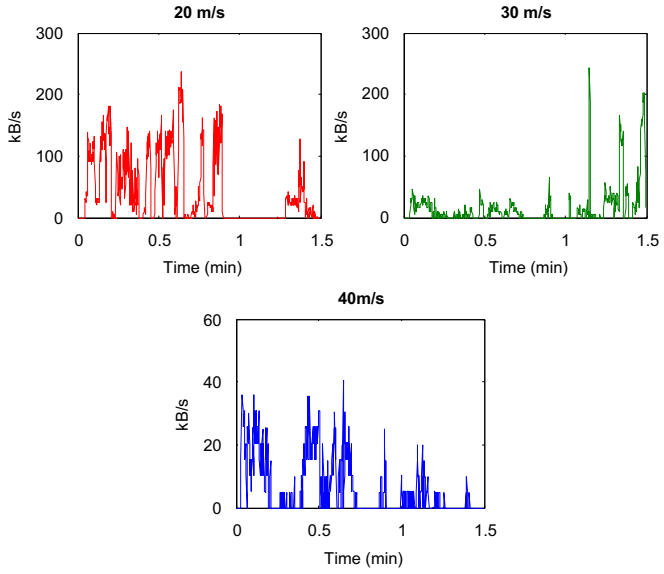

Fig. 12. Mean throughput of 12 vehicles at three different speeds.

that a fading radio propagation model, such as the Nakagami model, is best for the simulation of a vehicular environment (Taliwal et al., 2004).

Using this scenario as reference, we vary the number of vehicles and their speed in order to test the performance of COACH. Firstly, we show that COACH performs better than CRL in these scenarios. Then, we estimate the overhead that the usage of COACH introduces compared to using standard CRLs.

In this way, we start by running a simulation for 8 h, placing a single vehicle in the reference scenario. In this case, as there is full coverage and there are no other vehicles (i.e. channel contention is non-existent), the link to the infrastructure is quite stable. Figure 10 shows the obtained throughput for a single car at 72 km/h, which remains fairly constant at 250 kbps. Note that the VANETs are regarded as extreme environments that are opportunistically connected. Even if the average vehicle density is high, unbalanced traffic is inevitable and often leads to disconnection. In this simple scenario, the disconnection period is just of 15 min out of 8 h (3%) which can be considered as short disconnection period in such challenged networks. These 15 min of disconnection are due to 48 disconnection events, so that the mean time of disconnection is about 30 s per event. In our case this disconnection events are mainly due to handoffs between different RSUs. As the time required to query and retrieve a COACH response is less than 200 ms, vehicles have enough time to check the validity of a certificate under these circumstances.

Assuming that the CRL is encoded using some kind of Digital Fountain codes (Mackay, 2003, 2005; Luby, 2002) (e.g. Raptor codes Shokrollahi, 2006), the total number of pieces (N) that a vehicle must download to complete the CRL is $N=(1+\epsilon)M$ where M is the number of pieces in which the CRL is divided. Therefore, the size of the total number of pieces necessary to recover the CRL is slightly higher than the size of the original CRL. For instance, setting ϵ to 0.005, the amount of data necessary to recover a CRL for 50,000 vehicles is of approximately 146 MB.

Figure 11 shows the time that a single vehicle needs to download a CRL depending on the size of the CRL and its speed. In our scenario, where the CRL is expected to be around 145 MB, the approximate time to download is 9 min. In contrast, the time required to retrieved fresh certificate status information using COACH is less than 3 ms. So COACH is five orders of magnitude faster than a standard CRL when checking the status of a given certificate. Note that, in this case scenario, the speed of the vehicle is not really decisive, i.e. depending on the speed the time to download a CRL does not vary significantly.

Once we have checked that COACH works better than the standard CRL in the case of a single vehicle no matter its speed, we test COACH performance when there is more traffic in the road. To do so, we placed 12 vehicles in the reference scenario moving at different speeds.

Figure 12 shows the mean throughput obtained for vehicles at three different speeds. Contrary to the single vehicle scenario, in this case the throughput varies with time significantly. Notice that there are large periods of time during which the vehicles are not able to establish a link with the infrastructure. This is mainly due to the medium contention. Moreover, the throughput also shows different patterns depending on the vehicle's speed. Thus, faster vehicles are more prone to suffer from network disruptions than slower vehicles.

As a consequence to the decrement of the throughput at high speeds, the time to download a CRL increases. In this manner, vehicles at low speeds are able to download the CRL faster. This pattern is shown in Fig. 13. Moreover, it shows that slower vehicles are able to

Please cite this article as: Gañán C, et al. COACH: COllaborative certificate stAtus CHecking mechanism for VANETs. Journal of Network and Computer Applications (2012), doi:10.1016/j.jnca.2012.02.006

download bigger CRLs. In VANETS, where CRLs are expected to be of the order of hundreds of MB, only vehicles at 20 m/s are able to download the whole CRL. On the other hand, vehicles at 40 m/s are only able to download a piece of the CRL of 120 MB.

Table 4 shows the mean delays incurred when querying for the status of a given certificate. With *transmission delay* we denote the time to send the CSI query and the corresponding response. If we compare the transmission delay of the different revocation mechanisms, we can observe that ADOPT is the fastest but not so far from COACH. On the other hand, by *computational delay* we denote the time required to compose and validate a CSI response. In this case, ADOPT has the worst computational delay because each CSI response has to be signed by the CA. CRL computational delay is minimal as the CRL is only signed once and to searching the serial number of a certificate in the list has a computational cost of $O(\log_2 N)$. COACH only requires one CA signature but a $\mathcal{P}$ath has to be computed each time a CSI response is required, so the computational cost is similar to the CRL. Finally, we define Round-Trip Time (RTT) as the time that takes since a vehicle requests for CSI until the status of a given certificate is validated. Therefore, the RTT is affected by the transmission, computational and propagation delays. ADOPT has the worst RTT due to the multi-hop transmission of the cached CSI, while CRL and COACH download the CSI directly from the repository in range. In any case, the vehicles' speed affects transmission and RTT delays in all three revocation mechanisms.

Note that with these delays almost any envisioned application for VANET could timely perform the certificate validation. The most delay-constrained applications in VANETs are safety related (see Table 5). Since safety messages are sent out periodically by participating vehicles, in a dense network, a vehicle's onboard unit (OBU) may receive hundreds of such messages in a short time span. The ability to verify the status of these digitally signed messages quickly presents some challenges for OBUs since in order to keep the cost low, OBUs have limited computation power. COACH allows to quickly download fresh CSI and validate the status of a certificate. However, reaching the requirements of some safety applications such as "pre-crash sensing" is not always feasible. Vehicles going at 40 m/s or higher will not always be capable of validating the status of pre-crashing messages. In such situations, drivers going at such speeds must be aware of the risk associated with high speed. By contrast, data applications (such as peer-to-peer file sharing) are mostly delay-insensitive, and COACH will perform fine.

On the other hand, as it was shown in the previous section, depending on the amount of COACH queries that a vehicle performs, CRL can outperform COACH. However, a vehicle has to generate more than 200,000 COACH queries during the lifetime of the certificate status information (which is typically short) to create a load in the network similar to the one created by a CRL. Figure 14 shows the mean number of COACH queries that a

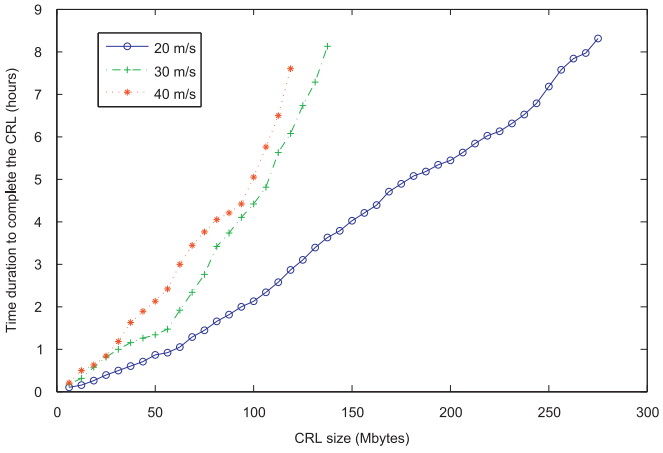

Fig. 13. Mean time to download a CRL for a twelve vehicle scenario.

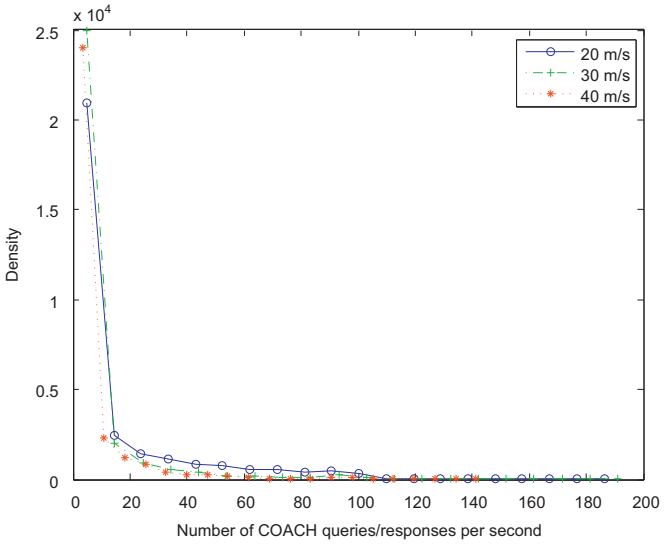

Fig. 14. Mean number of COACH queries per second.

Table 5
Representative applications.

Application	Communication requirements		
	Priority	Latency (ms)	Comm range (m)
Intersection collision warning	Class 1	~100	≤300
Pre-crash sensing	Class 1	~80	≤50
Transit vehicle signal priority	Class 2	~100	≤1000
Electronic toll collection	Class 3	~100	≤20
Internet access	Class 4	~500	≤300
Roadside service finder	Class 4	~500	≤300

Table 4
Delays when querying for CSI.

Vehicle speed (m/s)	Delay								
	COACH			CRL			ADOPT		
	Transmission (ms)	Computational (ms)	RTT (ms)	Transmission (h)	Computational (ms)	RTT (h)	Transmission (ms)	Computational (ms)	RTT (ms)
20	75	2401	78	3521	2400	3521	72	3612	101
30	149	2401	157	8213	2400	8214	122	3600	312
40	173	2401	187	9811	2400	9813	152	3600	421

Please cite this article as: Gañán C, et al. COACH: COllaborative certificate stAtus CHecking mechanism for VANETs. Journal of Network and Computer Applications (2012), doi:10.1016/j.jnca.2012.02.006

vehicle could generate per second. Note that almost in any case, a vehicle can receive a COACH response per second, and in some cases it can receive more than 180 responses per second. In this sense, a vehicle at 20 m/s can receive an average of 14 responses per second, while a vehicle at 40 m/s reduces the mean number of responses per second to 6. Therefore, any vehicle independently of its speed is able to check the status of at least half dozen of certificate per second. On the other hand, this experimental results show that a vehicle could generate more than 200,000 COACH queries during long road trips. In these cases, COACH will degrade the bandwidth efficiency while still being better in terms of availability than CRL.

Finally, in order to extend the results obtained for the 12-vehicle scenario to a more crowded scenario, we double the number of vehicles that operate in the reference scenario. With more traffic in the network, as expected, we will see that COACH outperforms CRL even more clearly.

Figure 15 shows the throughput obtained for three vehicles at different speeds. Similarly to the 12-vehicle scenario, the throughput varies with time significantly. Again, there are large periods of time during which the vehicles are not able to establish a link with the infrastructure. Hence, faster vehicles are more prone to suffer from network disruptions than slower vehicles. Note that the throughput of any vehicle at 40 m/s is nearly zero during large periods of time, i.e. fast vehicles will not be able to download big files such as a CRL.

In this way, Fig. 16 shows the mean time to download a CRL in this scenario. Contrary to the 12-vehicle scenario, no vehicle is able to download a CRL of 146 MB during the 8 h that the simulation lasted. The maximum amount of information that a vehicle is able to download during this amount time is 69 MB, and it is only achieved by slower vehicles. Therefore, with 24 vehicles the amount of time that a vehicle takes to download a CRL is unfeasible for the proper performance of the PKI. It is not viable that a vehicle spend more than 8 h to download fresh certificate status information.

Therefore, it becomes necessary to use another mechanism to distribute the certificate status information. Using COACH a vehicle at 20 m/s can receive an average of 4 responses per second, and a vehicle at 40 m/s can receive 2 responses per second (see Fig. 17). That means that in the worst case scenario a car can check at least 2 certificates per second in average.

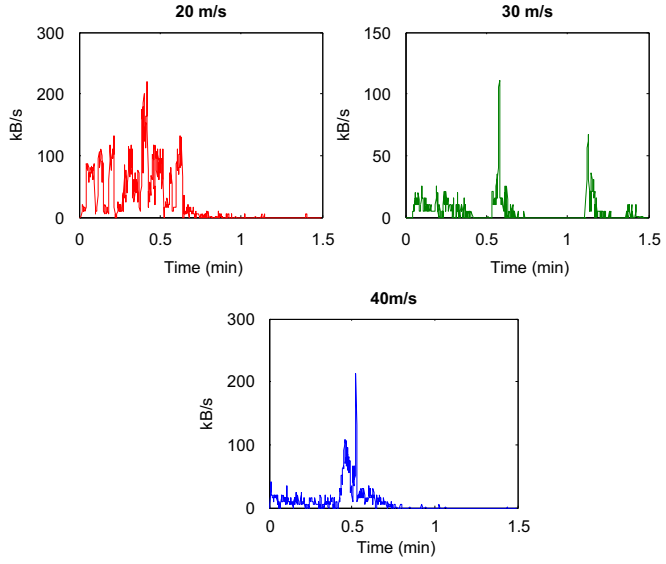

Fig. 15. Throughput of 24 vehicles at three different speeds.

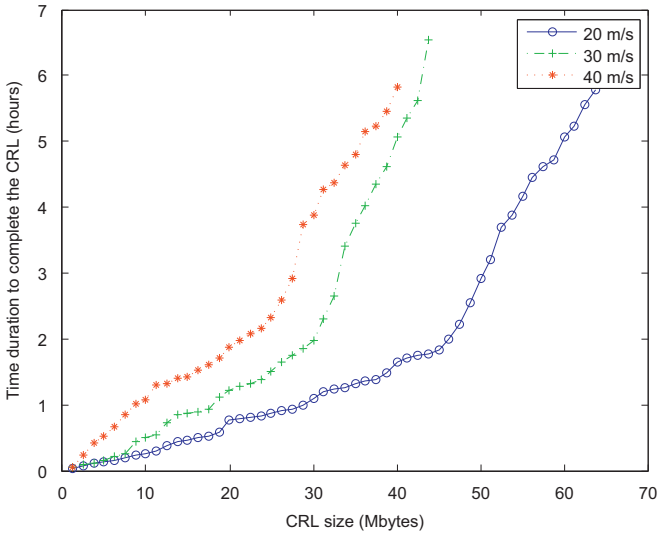

Fig. 16. Mean time to download a CRL for a 24-vehicle scenario.

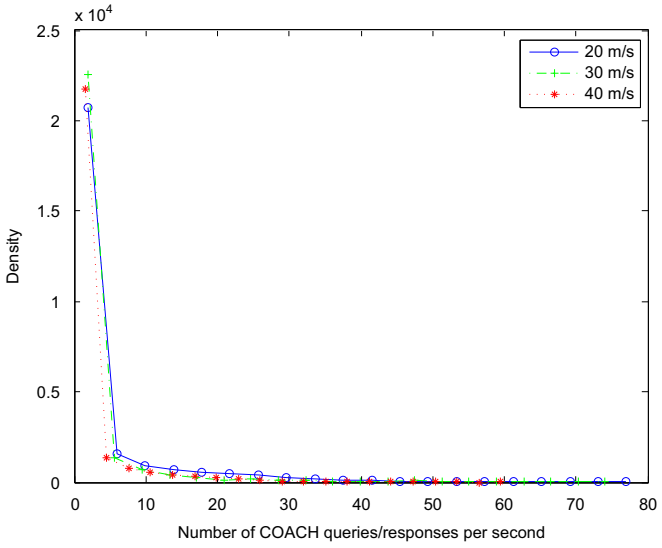

Fig. 17. Mean number of COACH queries per second.

In conclusion, COACH works better than CRL in terms of bandwidth efficiency no matter the vehicle speed. In the following we show the way EvCOACH improves COACH performance. In terms of network overhead, COACH and EvCOACH are very similar as they only differ in 15 B in the response size. Therefore, EvCOACH will also work better than CRL in the same way COACH does.

Besides the network load decrease that COACH induces in these scenarios, EvCOACH also enhances the certificate validation workload when there are no new revoked certificates. As explained in Section 4, by including a nonce in the responses, COACH revalidates previous cached certificate status information. Notice that a vehicle does need to download again the CRL and only needs to check the freshness and authenticity of the nonces. Table 6 shows the savings in time and bandwidth that EvCOACH provides when using the revalidation of the COACH tree during consecutive CSI updates. Note that the table shows the best case scenario when a vehicle is able to download a CRL in 3.5 h. However, the benefits of using this mechanism are expected to be higher in a real scenario where vehicles will spend more than 8 h to download a 146 MB CRL.

Please cite this article as: Gañán C, et al. COACH: COllaborative certificate stAtus CHecking mechanism for VANETs. Journal of Network and Computer Applications (2012), doi:10.1016/j.jnca.2012.02.006

Table 6
Network bandwidth and time saving of EvCOACH during consecutive CSI updates.

Update index	$d=1$	$d=2$	$d=3$
Network bandwidth saving (MB)	~146	~292	~438
Time saving (h)	~3.5	~7	~10.5

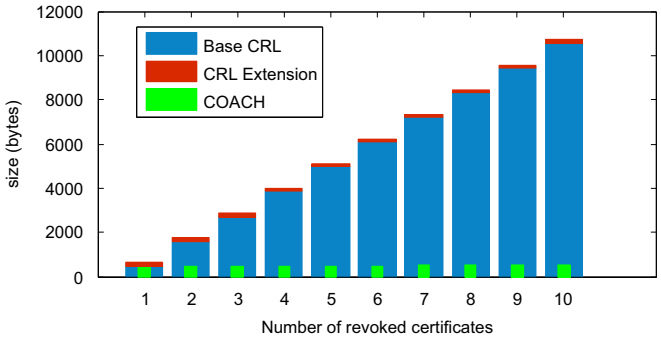

Fig. 18. Overhead introduced by the *extended-CRL* (using ECDSA-256).

At this point, we have shown that COACH and EvCOACH performance surpasses standard CRL in the simulated scenarios in terms of CSI availability. Henceforward, we evaluate the overhead that COACH introduces to the network. As explained in the previous section, COACH introduces an extension to the standard CRL in order to allow any vehicle and RSU to become a repository. Figure 18 shows the overhead introduced by the *extended-CRL*.

Note that the overhead introduced is independent of the number of revoked certificates, so that in our scenario with 50,000 vehicles the introduced overhead represents a 0.011065% of the total size of the *extended-CRL*. Therefore, we can conclude that this overhead is negligible in a VANET scenario. Additionally, Fig. 18 shows that the size of the COACH responses grows much slower than the size of the CRL.

7. Conclusions

Local revocation approaches based on threshold cryptography and voting schemes provide mechanisms for revocation management inside the VANET. However, the local validity of the certificate status information and the lack of support for extending its validity to the global network restrain their utilization in the real VANET scenarios. These problems can be mitigated by adapting traditional PKI to the vehicular environment. Standard certificate validation mechanisms, such as CRL or OCSP, do not fit well in a VANET where huge number of nodes are involved and where several pseudonym certificates are assigned in addition to vehicle identity certificates.

In this paper, we have presented COACH, a collaborative certificate status checking mechanism based on an *extended-CRL*. The main advantage of this *extended-CRL* is that the roadside units and repository vehicles can build an efficient structure based on an authenticated hash tree to respond to status checking requests inside the VANET, saving time and bandwidth. In addition, we have described EvCOACH, an extension of the COACH mechanism that improves the performance of the standard protocol by avoiding to reissue useless certificate status information. In this context, EvCOACH allows to revalidate a previously downloaded CRL so that RSUs and repository vehicles reduce the computational cost of building the hash tree.

Analytical results show that allocating a small bandwidth is enough to ensure that vehicles receive certificate status responses within few seconds. The performance improvement is obtained at expenses of adding the signed hash tree extension to the standard-CRL. In this way, COACH becomes an offline certificate status validation mechanism as it does not need trusted responders to operate. Therefore, COACH significantly reduces the complexity of certificate management and achieves great efficiency and scalability, particularly when it is deployed in heterogeneous vehicular networks.

Acknowledgments

This work has been supported partially by the Spanish Research Council with Project (TEC2008-06663-C03-01), by Spanish Ministry of Science and Education with Project CONSOLIDER CSD2007-00004 (ARES), and SERVET (TEC2011-26452) and by Generalitat de Catalunya with Grant 2009 SGR-1362 to consolidated research groups. We would like also to thank the support and reviews of our friends Juan Miguel and Sergi Reńe.

References

Abrougui K, Boukerche A. Secure service discovery protocol for intelligent transport systems: proof of correctness. In: Proceedings of the 1st ACM international symposium on design and analysis of intelligent vehicular networks and applications (DIVANet'11); 2011. p. 101–8.

Abrougui K, Boukerche A, Pazzi R. Location-aided gateway advertisement and discovery protocol for VANETs. IEEE Transactions on Vehicular Technology 2010;59:3843–58.

Armknecht F, Festag A, Westhoff D, Zeng K. Cross-layer privacy enhancement and non-repudiation in vehicular communication. In: 4th workshop on mobile ad hoc networks (WMAN'07); 2007.

Bera R, Bera J, Sil S, Dogra S, Sinha N, Mondal D. Dedicated short range communications (DSRC) for intelligent transport system. In: IFIP international conference on wireless and optical communications networks; 2006. p. 1–5.

Berkovits S, Chokhani S, Furlong J, Geiter J, Guild J. Public key infrastructure study: final report. Technical Report, MITRE Corporation for NIST; 1995.

Chen W, Guha R, Chennikara-Varghese J, Pang M, Vuyyuru R, Fukuyama J. Context-driven disruption tolerant networking for vehicular applications. In: IEEE vehicular networking conference (VNC); 2010. p. 33–40.

Fan C-I, Hsu R-H, Tseng C-H. Pairing-based message authentication scheme with privacy protection in vehicular ad hoc networks. In: Proceedings of the international conference on mobile technology, applications, and systems, mobility '08; 2008. p. 82:1–7.

Forné J, Muñoz JL, Esparza O, Hinarejos F. Certificate status validation in mobile ad hoc networks. Wireless Communications 2009;16:55–62.

Haas J, Hu Y-C, Laberteaux K. Efficient certificate revocation list organization and distribution. IEEE Journal on Selected Areas in Communications 2011;29:595–604.

Haas JJ, Hu Y-C, Laberteaux KP. Design and analysis of a lightweight certificate revocation mechanism for VANET. In: Proceedings of the 6th ACM international workshop on VehiculAr InterNETworking (VANET'09). New York, NY, USA: ACM; 2009. p. 89–8.

Hubaux J, Capkun S, Luo J. The security and privacy of smart vehicles. IEEE Security Privacy 2004;2:49–55.

IEEE. IEEE trial-use standard for wireless access in vehicular environments—security services for applications and management messages. IEEE Std 1609.2-2006; 2006. p. 1–117.

Jain S, Fall K, Patra R. Routing in a delay tolerant network. In: Proceedings of the 2004 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM'04); 2004. p. 145–8.

Jiang D, Delgrossi L. IEEE 802.11p: towards an international standard for wireless access in vehicular environments. In: IEEE vehicular technology conference (VTC Spring 2008); 2008. p. 2036–40.

Kim J, Baek J, Kim K, Zhou J. A privacy-preserving secure service discovery protocol for ubiquitous computing environments. In: Proceedings of the 7th European conference on public key infrastructures, services and applications (EuroPKI'10); 2011. p. 45–60.

Krajzewicz D, Hertkorn G, Rössel C, Wagner P. Sumo (simulation of urban mobility); an open-source traffic simulation. In: 4th middle east symposium on simulation and modelling (MESM2002); 2002. p. 183–7.

Laberteaux KP, Haas JJ, Hu Y-C. Security certificate revocation list distribution for VANET. In: Proceedings of the 5th ACM international workshop on VehiculAr Inter-NETworking (VANET'08); 2008. p. 88–9.

Lin X, Lu R, Zhang C, Zhu H, Ho P-H, Shen X. Security in vehicular ad hoc networks. IEEE Communications Magazine 2008;46:88–95.

Lu R, Lin X, Zhu H, Ho P-H, Shen X. Ecpp: efficient conditional privacy preservation protocol for secure vehicular communications. In: The 27th IEEE conference on computer communications (INFOCOM 2008); 2008. p. 1229–7.

Luby M. Lt codes. In: Proceedings of the 43rd annual IEEE symposium on foundations of computer science, 2002; 2002. p. 271–80.

Ma C, Hu N, Li Y. On the release of crls in public key infrastructure. In: Proceedings of the 15th conference on USENIX security symposium, vol. 15. Berkeley, CA, USA: USENIX Association; 2006.

Mackay DJC. Information theory, inference and learning algorithms; 2003.

Mackay DJC. Fountain codes. IEE Proceedings Communications 2005;152:1062–8.

Marias Giannis F., Papapanagiotou Konstantinos, Tsetsos Vassileios, Sekkas Odysseas, Georgiadis Panagiotis. Integrating a trust framework with a distributed certificate validation scheme for MANETs. EURASIP Journal of Wireless Communications and Networking. 2006: 2(April 2006): 77. doi:10.1155/WCN/2006/78259. 〈http://dx.doi.org/10.1155/WCN/2006/78259〉.

Marias GF, Papapanagiotou K, Georgiadis P. Adopt. A distributed OCSP for trust establishment in MANETs. In: 11th European wireless conference 2005; 2005.

McCanne S, Floyd S. The Network Simulator NS-2 〈http://www.isi.edu/nsnam/ns/〉.

Merkle R. A certified digital signature. In: Advances in cryptology (CRYPTO89). Lecture notes in computer science, vol. 435. Springer-Verlag; 1989. p. 234–6.

Merkle R. A certified digital signature. In: Advances in cryptology CRYPTO'89 proceedings. Lecture notes in computer science, vol. 435. Berlin/Heidelberg: Springer; 1990. p. 218–8.

Moschetta E, Antunes RS, Barcellos MP. Flexible and secure service discovery in ubiquitous computing. Journal of Network and Computer Applications 2010; 33:128–40.

Myers M, Ankney R, Malpani A, Galperin S, Adams C. X.509 Internet public key infrastructure online certificate status protocol (OCSP). RFC 2560 Internet engineering task force; 1999.

Nowatkowski M, Wolfgang J, McManus C, Owen H. The effects of limited lifetime pseudonyms on certificate revocation list size in VANETs. In: Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon); 2010. p. 380–3.

Papadimitratos P, Buttyan L, Holczer T, Schoch E, Freudiger J, Raya M, et al. Secure vehicular communication systems: design and architecture. IEEE Communications Magazine 2008a;46:100–9.

Papadimitratos P, Buttyan L, Hubaux J-P, Kargl F, Kung A, Raya M. Architecture for secure and private vehicular communications. In: 7th international conference on ITS telecommunications, 2007 (ITST'07); 2007. p. 1–6.

Papadimitratos P, Mezzour G, Hubaux J-P. Certificate revocation list distribution in vehicular communication systems. In: Proceedings of the 5th ACM international workshop on VehiculAr Inter-NETworking (VANET'08); 2008b. p. 86–7.

Raya M, Hubaux J-P. The security of vehicular ad hoc networks. In: Proceedings of the 3rd ACM workshop on security of ad hoc and sensor networks (SASN'05); 2005. p. 11–21.

Reñé S, Mata J, Alins J, Hernadez C, Muñoz-Tapia JL, Óscar Esparza, et al. VANET emulator platform; 2010 〈http://anka.upc.es〉.

Reñé S, Gañán C, Caubet J, Alins J, Mata J, Muñoz JL. Analysis of video streaming performance in vehicular networks. In: The 1st international conference on advanced communications and computation. Barcelona, Spain; 2011.

Shokrollahi A. Raptor codes. IEEE/ACM Transactions on Networking 2006;14: 2551–67.

Taliwal V, Jiang D, Mangold H, Chen C, Sengupta R. Empirical determination of channel characteristics for dsrc vehicle-to-vehicle communication. In: Proceedings of the 1st ACM international workshop on vehicular ad hoc networks (VANET'04). New York, NY, USA: ACM; 2004. p. 88.

Walleck D, Li Y, Xu S. Empirical analysis of certificate revocation lists. In: Proceedings of the 22nd annual IFIP WG 11.3 working conference on data and applications security; 2008. p. 159–4.

Wasef A, Jiang Y, Shen X. DCS: an efficient distributed-certificate-service scheme for vehicular networks. IEEE Transactions on Vehicular Technology 2010; 59:533–49.

Wasef A, Shen X. Maac: Message authentication acceleration protocol for vehicular ad hoc networks. In: IEEE global telecommunications conference, 2009 (GLOBECOM 2009); 2009. p. 1–6.

Zhang C, Lu R, Lin X, Ho P-H, Shen X. An efficient identity-based batch verification scheme for vehicular sensor networks. In: The 27th IEEE conference on computer communications (INFOCOM 2008); 2008. p. 246 –50.

Please cite this article as: Gañán C, et al. COACH: COllaborative certificate stAtus CHecking mechanism for VANETs. Journal of Network and Computer Applications (2012), doi:10.1016/j.jnca.2012.02.006