MHT-Based Mechanism for Certificate Revocation in VANETs

Jose L. Muñoz[1(✉)], Oscar Esparza[1], Carlos Gañán[1], Jorge Mata-Díaz[1], Juanjo Alins[1], and Ivan Ganchev[2]

[1] Departament Enginyeria Telemàtica,
Universitat Politècnica de Catalunya, Barcelona, Spain
{jose.munoz,oscar.esparza,carlos.ganan,jmata,juanjo}@entel.upc.es
[2] Telecommunications Research Centre,
University of Limerick, Limerick, Ireland
ivan.ganchev@ul.ie

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 utilized to provide these functionalities using digital certificates. However, if a vehicle is no longer trusted, its certificates have to be immediately revoked and this status information has to be made available to other vehicles as soon as possible. The goal of this chapter is to introduce and describe in detail a certificate revocation mechanism based on the Merkle Hash Tree (MHT), which allows to efficiently distribute certificate revocation information in VANETs. For this, an extended-CRL is created by embedding a hash tree in each standard certificate revocation list (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. This means that any node possessing an extended-CRL, including Road Side Units (RSUs) or intermediate vehicles, can produce short certificate-status responses that can be easily authenticated. The main procedures involved in the proposed mechanism are described in detail. General security issues related to the mechanism are treated as well.

Keywords: PKI · Certificate revocation · Extended-CRL · MHT · VANETs

1 Introduction

In the last years, wireless communications between vehicles 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 AdHoc Networks (VANETs), which consist of mobile nodes capable of communicating with each other (Vehicle to Vehicle communication,V2V) and with the infrastructure (Vehicle to Infrastructure communication,V2I). To make this new type of communications feasible,

I. Ganchev et al. (Eds.): Wireless Networking for Moving Objects, LNCS 8611, pp. 282–300, 2014.
DOI: 10.1007/978-3-319-10834-6_15

vehicles are equipped with On-Board Units (OBUs). In addition, fixed communication units Road Side Units (RSUs) are placed along the road. Finally, multihop communication based on the IEEE802.11p standard is utilized to facilitate data exchange among network nodes that are not within radio range of each other [1,2].

The open-medium nature of these networks makes it necessary to integrate VANET security mechanisms such as authentication (the assurance to one entity that another entity is who it claims to be), message integrity (the assurance to an entity that data has not been altered), confidentiality (the assurance to an entity that no one can read a particular piece of data except the receiver explicitly intended) and privacy (the quality of being secluded from the presence or view of others) [3].

A generic solution envisioned to achieve these functionalities is based on digital certificates provided by a centralized Certification Authority (CA) [4,5]. This is the approach of the IEEE 1609.2 standard [6], which states that certificates will be used to digitally sign and encrypt the exchanged messages by using the Elliptic Curve Integrated Encryption Scheme (ECIES). Therefore, vehicular networks will rely on a Public Key Infrastructure (PKI) for certificate management. The possibility to assign in the future a personal IPv6 address [7] to each vehicle and embed this address in the vehicle's certificate is also envisaged as a promising proposal for solving many of the security issues existing in these networks.

On the other hand, a critical part of the PKI is the certificate revocation. In general, the revocation approaches for VANETs can be roughly classified as global or local:

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

According to the IEEE 1609.2 standard [6], vehicular networks will utilize the global revocation approach based on PKI Certificate Revocation Lists (CRLs). 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 networks contain many nodes (vehicles) and also because each vehicle will probably have many pseudonyms[1] to protect the users' privacy. As a result, a VANET CRL might have a size of hundreds of Megabytes [8,9]. The distribution of such a huge data structure within a VANET is a challenging issue that has attracted the attention of many researchers [3,8,10,11]. A general conclusion is that most of the research efforts

[1] Certificates with pseudonyms contain a name that a the user assumes for operating in the VANET but that differs from his or her original or true name. These certificates are temporary.

have been put into trying to reduce the size of the CRL, either by splitting or compressing (c.f. Sect. 3).

This chapter explains in detail another approach for reducing the amount of certificate-revocation data, exchanged between VANET nodes. The mechanism described is an adaptation of previous works [12,13] for the VANET scenario. The proposed certificate revocation mechanism is based on the Merkle Hash Tree (MHT) and a model of computation, whereby untrusted repositories answer certificate status queries on behalf of the CA and provide a proof of the validity of the response back to the user. By using this mechanism, two major CRL issues can be tackled: (i) the CA is no longer a bottleneck as there are several repositories that act on its behalf; and (ii) the revocation data can be checked without downloading the whole CRL.

The main idea behind the proposal described in this chapter is to embed a little extra information into the CRL which allow the efficient and timely checking/obtaining the status of a certificate without having to download the entire CRL. This information is not proportional to the number of certificates but it roughly occupies the size of a signed hash value and the fixed overhead of introducing a CRL extension.

More specifically, this chapter proposes a way of efficiently embedding a MHT within the structure of the standard CRL to generate a so-called extended-CRL. For the creation of the extended-CRL, a standard way of adding extra information to the CRL is used. This extension contains all the necessary information to allow any vehicle, or VANET infrastructure element that possesses the extended-CRL, to build the MHT, i.e., a hash tree with the certificate status information of the CRL. Using this MHT tree, any entity possessing the extended-CRL can act as a repository and efficiently answer certificate status checking requests of other vehicles or VANET nodes.

In general, the response with certificate status information is short in size (less than 1 KB), which allows to perfectly fit it within a single UDP message. This makes the distribution of certificate status information more efficient than distributing the whole CRLs (even if they are compressed), thus reducing the amount of data that has to be transmitted over the VANET. The proposed certificate revocation mechanism operates off-line, i.e. no on-line trusted entity (like a CA) is needed for authenticating the responses produced by the repositories (c.f. next section). However, repositories need to have access from time to time to the CA to download the CRL.

The rest of the chapter is organized as follows. Section 2 presents a brief definition of the certificate revocation paradigm along with the corresponding classification of the certificate status checking mechanisms. Section 3 describes the existing global revocation proposals for VANETs. Section 4 explains the operation of the hash tree. Section 5 describes the proposed certificate revocation mechanism and the various procedures involved in it. Section 6 discusses general security issues related to this mechanism. Finally, Sect. 7 concludes the chapter.

2 Certificate Revocation Paradigm

The owner of the certificate to be revoked, an authorized representative or the issuer CA, can initiate the revocation process for this certificate (Fig. 1).

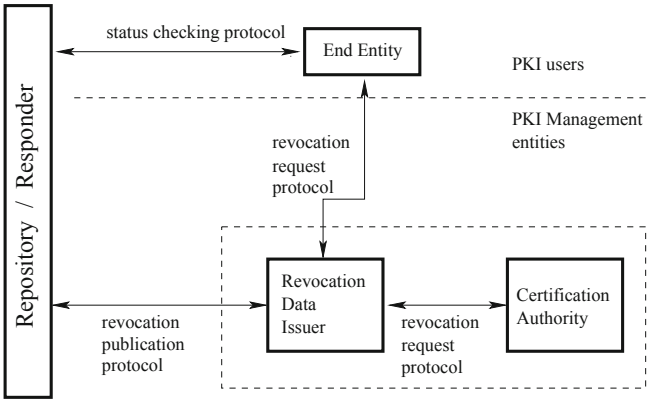

Fig. 1. PKI reference model

To revoke the certificate, any of the authorized entities generates a revocation request and sends it to the Revocation Data Issuer (RDI). The revocation data issuer is a Trusted Third Party (TTP) that has the master database of revoked certificates. The revocation data issuer is also responsible for transforming its database revocation records into "certificate status data". The status data has the appropriate format to be distributed to the end entities and includes at least the following data:

- A *Certificate Issuer* which is the Distinguished Name (DN) of the CA that issued the target certificate(s).
- A *Validity Period* that is the life-time of the status data. Obviously this validity period is much smaller than the validity period of the certificate.
- A *Issuer Name* that is the DN of the TTP that issued the status data.
- A *Cryptographic Proof* that must demonstrate that the status data was issued by a TTP.
- A *Serial Number* of the target certificate(s).
- A *Revocation Date* that is the date when the target certificate was revoked.
- A *Revocation Reason* which is optional. This is used guidance and it can be `unspecified`, `keyCompromise`, `cACompromise`, `affiliationChanged`, `superseded`, `removeFromCRL`, `cessationOfOperation` or `certificateHold`.

In the vast majority of the revocation systems, end entities do not have a straight connection to the revocation data issuer. Instead, the revocation data issuer publishes the status data in "repositories" or "responders". The main function of both repositories and responders is to answer requests from end entities concerning the status of certificates (status checking). The difference between them is that the repositories are non-TTPs that store status data pre-computed by the revocation data issuer while the responders are TTPs that have a private key and can provide a signature (serving as a cryptographic proof) for each response. Maintaining the level of security is one of the main drawbacks of using responders, in the sense that the responder has to be on-line, but at the same time it has to protect its private key against intruders. Certificate status checking mechanisms can be classified in different ways [14]:

1. By the type of certificate status checking mechanism:
   - In an *off-line mechanism* the status data is pre-computed by a revocation data issuer and then it is distributed to the requester by a repository.
   - In an *on-line mechanism* the status data is provided on-line by a responder and a cryptographic proof is generated for each request. This provides up-to-date information.
2. By the type of list:
   - *Negative or black lists* contain revoked certificates.
   - *Positive or white lists* contribute valid certificates.
3. By the way of providing evidence:
   - A *direct evidence* is given if a certificate is mentioned in a positive or negative list, respectively. Then, accordingly, it is supposed to be either revoked or not.
   - An *indirect evidence* is given if a certificate cannot be found on a list and therefore, the contrary is assumed.
4. By the way of distributing information:
   - In a *push mechanism* the repository or the responder periodically sends updates to its users.
   - In a *pull mechanism* the user asks the repository or the responder for certificate status data.

It is worth mentioning that certificate status checking is the mechanism that has the greatest impact on the overall performance of a certificate revocation system. Therefore, a certificate status checking needs to be fast, efficient and timely, and it must scale well too. It is therefore necessary to reduce the number of time-consuming calculations like generation and verification of digital signatures, and to minimize the amount of data transmitted.

3 Related Work

This section describes the existing global revocation proposals for VANETs.

Overview of Centralized Revocation Approaches. The IEEE 1609.2 standard [6] proposes an architecture based on the existence of a TTP, which manages the revocation service. In this architecture, each vehicle possesses several short-lived certificates (used as pseudonyms), to ensure users' privacy. However, short-lived certificates are not sufficient because compromised or faulty vehicles could still endanger other vehicles until the end of their certificate lifetimes. Thus, the IEEE 1609.2 promotes the use of CRLs to manage revocation while assuming a pervasive roadside architecture.

Other proposals in the literature also assume the existence of a TTP to provide the revocation service. Raya *et al.* [3] propose the use of a tamper-proof device[2] to store the certificates. A TTP is in charge of pre-loading the cryptographic material in the tamper-proof device. Thus, when a vehicle is compromised/misbehaving, it can be removed from the network by just disabling its tamper-proof device. To that end, the TTP must include the corresponding revocation information in a CRL. To reduce the bandwidth consumed by the transmission of CRLs, the authors in [3] proposed to compress the CRLs by using Bloom filters[3]. However, this method gives rise to false positives which degrades the reliability of the revocation service.

On the other hand, even compressed, the timely distribution of CRLs to all vehicles is not a trivial process. Some authors [5,10], instead of using a single central authority, have proposed the use of regional certification authorities with developed trust relationships. Papadimitratos *et al.* [15] suggest restricting the scope of the CRL within a region. Visiting vehicles from other regions are required to obtain temporary certificates. Thus, a vehicle will have to acquire temporary certificates if it is traveling outside its registered region. The authors also propose breaking the CRL into different pieces, then transmitting these pieces using Fountain or Erasure codes, so that a vehicle can reconstruct the CRL after receiving a certain number of pieces. Similarly, in [16], 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 when RSUs do not completely cover the domain of a CA, V2V communications are used to distribute the CRL to all the vehicles [11]. This mechanism is also used in [17,18], where it is detailed as a PKI mechanism based on bilinear mapping. Revocation is accomplished through the distribution of CRL that is stored by each user.

Overview of Decentralized Revocation Approaches. Decentralized revocation mechanisms provide the revocation service without assuming the existence of a TTP. Some proposals in the literature divert from the IEEE 1609.2 standard and use on-line certificate status checking protocols instead of CRLs to provide a revocation service in a decentralized manner. This is the case, of the Ad-hoc

[2] Tamper-proof devices are designed to resist intentional malfunction or sabotage by any user with physical access to the device.

[3] A Bloom filter is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set.

Distributed OCSP for Trust (ADOPT) [19], which uses cached OCSP responses that are distributed and stored on intermediate nodes. Another group of proposals establishes the revocation service on detecting a vehicle to be misbehaving by a set of other vehicles. Then, the detecting set may cooperatively revoke the credential of the misbehaving node from their neighborhood. Moore *et al.* proposed in [20] a revocation mechanism aiming to prevent an attacker from falsely voting against legitimate nodes. Raya *et al.* in [3] proposed a mechanism to temporarily remove an attacker from the trust list if the CA is unavailable. To do so, the number of accusing neighbor users must exceed a threshold. A similar mechanism based also on vehicle voting is proposed in [21]. Again, by means of a voting scheme, a vehicle can be marked as misbehaving and then removed by its neighbors from the trust list.

Another proposal uses a game-theoretic revocation approach to define the best strategy for each individual vehicle [22]. These mechanisms provide incentives to guarantee the successful revocation of the malicious nodes. Moreover, thanks to the records of past behavior, the mechanism is able to dynamically adapt the parameters to nodes' reputations and establish the optimal Nash equilibrium on-the-fly, minimizing the cost of the revocation.

Finally, there are some hybrid approaches that are neither totally centralized nor decentralized [23–27]. For instance, authors in [28] propose the use of authenticated data structures to issue the certificate status information. Using these schemes, the revocation service is decentralized to transmit the certificate status information but still depends on a CA to decide when a node should be evicted from the VANET.

4 Operation of the Hash Tree

The Merkle Hash Tree (MHT) [29] relies on the properties of the one way hash functions. MHT exploits the fact that a one way hash function is at least 10,000 times faster to compute than a digital signature, so the majority of the cryptographic operations performed in the revocation system are hash functions instead of digital signatures. A sample MHT is presented in Fig. 2.

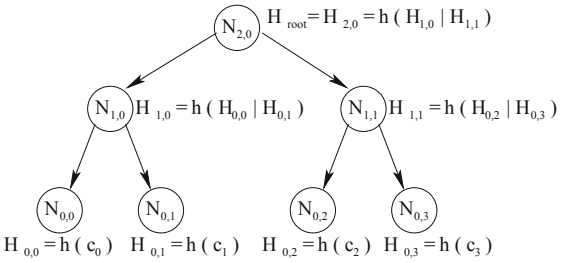

Fig. 2. Sample MHT.

$N_{i,j}$ denotes the j-th node at the i-th level. $H_{i,j}$ denotes the cryptographic variable 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 have been revoked,

$$\Phi = \{c_0, c_1, \ldots, c_j, \ldots, c_n\}, \tag{1}$$

where c_j is the data stored by leaf $N_{0,j}$. Then, $H_{0,j}$ is computed as

$$H_{0,j} = h(c_j), \tag{2}$$

where h is a one way hash function.

To build the MHT, a set of t adjacent nodes at a given level i (i.e. $N_{i,j}$, $N_{i,j+1} \ldots, N_{i,j+t-1}$), are combined into one node in the upper level, denoted by $N_{i+1,k}$. Then, $H_{i+1,k}$ is obtained by applying h to the concatenation of the t cryptographic variables:

$$H_{i+1,k} = h(H_{i,j} | H_{i,j+1} | \ldots | H_{i,j+t-1}) \tag{3}$$

At the top level, there is only one node called the "root". H_{root} is a digest for all the data stored in the MHT.

The sample MHT in Fig. 2 is a binary tree because adjacent nodes are combined in pairs to form a node in the next level ($t = 2$) and $H_{root} = H_{2,0}$.

Definition 1. The $\mathcal{D}$igest is defined as
$\mathcal{D}igest = \{DN_{RDI}, H_{root}, Validity\ Period\}_{SIG_{RDI}}$

Definition 2. The $\mathcal{P}$ath$_{c_j}$ is defined as the set of cryptographic values necessary to compute H_{root} from the leaf c_j.

Remark 1. Note that the $\mathcal{D}$igest is trusted data because it is signed by the revocation data issuer and it is unique within the tree, while $\mathcal{P}$ath is different for each leaf.

Claim. If the MHT provides a response with the proper $\mathcal{P}$ath$_{c_j}$ and the MHT $\mathcal{D}$igest, an end entity can verify whether $c_j \in \Phi$.

Example 1. Let's suppose that a certain user wants to find out whether c_1 belongs to the sample MHT in Fig. 2. Then,
$\mathcal{P}$ath$_{c1} = \{N_{0,0}, N_{1,1}\}$
$\mathcal{D}igest = \{DN_{RDI}, H_{2,0}, Validity\ Period\}_{SIG_{RDI}}$
The response verification consists in checking that $H_{2,0}$ computed from the $\mathcal{P}$ath$_{c_1}$ matches $H_{2,0}$ included in the $\mathcal{D}$igest:

$$H_{root} = H_{2,0} = h(h(h(c_1)|H_{0,0})|H_{1,1}) \tag{4}$$

Remark 2. Note that the MHT can be built by a TTP (revocation data issuer) and distributed to a repository because a leaf cannot be added or deleted to Φ without modifying H_{root}[4] which is included in the $\mathcal{D}$igest, and as the $\mathcal{D}$igest is signed, it cannot be forged by a non-TTP.

[4] To do this, an attacker needs to find a pre-image of a one way hash function which is computationally infeasible by definition.

5 MHT-Based Mechanism for Certificate Revocation in VANETs

5.1 Overview

Our mechanism is a centralized revocation system based on an adaptation of the typical PKI CRL for the vehicular environment. We use a CRL extension to embed a Merkle Hash Tree, which allows us to check certificate status data without downloading the whole CRL.

The mechanism is implemented over a hierarchical architecture that consists of three levels (Fig. 3): the CA is located at level 1, the RSUs are located at level 2, and the OBUs are located at level 3, the bottom of the hierarchy. The main tasks of each entity are presented below:

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 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 on the infrastructure side, which does not suffer from disconnections. If the CA considers that an RSU has been compromised, the CA can exclude it from the trust list.
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. Regarding the design of the revocation system, the main issue to address is that the transmission rate between the OBUs and the RSUs for transferring certificate status data might be a bottleneck.

The proposed certificate revocation mechanism consists of 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 third non-trusted parties to answer certificate status checking requests in an off-line 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 certificate status information from an available VANET repository. The *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 the certificate status requests.

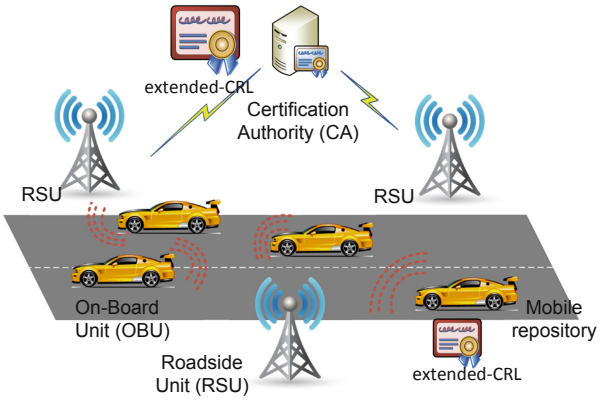

Fig. 3. System Architecture.

The steps followed by the CA are described below:

1. Create a *tbs-CRL* (to be signed CRL), which 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, time-stamps to establish the validity period, etc.
2. Create the MHT tree, that is, a MHT that is constructed by using the serial numbers within the previous *tbs-CRL* as leaves of the tree.
3. Calculate the extension, which consists basically of the $\mathcal{D}$igest. Once calculated, append the $\mathcal{D}$igest to the tbs-CRL, generating the *tbs-extended-CRL*. Just recall that this $\mathcal{D}$igest is calculated as the concatenation of the certification authority distinguished number, the root hash and the validity period of the certificate status information, and after that it is 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 MHT tree is just a different way of representing the certificate status information, but the hash tree will be valid during the same time and will provide the same information than the CRL.
4. Sign the tbs-extended-CRL, generating the *extended-CRL*. Note that this second overall signature not only authenticates all the certificate status information, but also binds this certificate status information to the $\mathcal{D}$igest. The *extended-CRL* is only slightly larger than the standard CRL.
5. Distribute copies of the *extended-CRL* to the designated RSUs which are the repositories.

5.2 Responding to Certificate-Status Requests

The MHT embedded in the CRL will help us to efficiently respond to certificate-status requests. Table 1 summarizes the information contained in each leaf of the MHT.

Table 1. Leaf Information.

left child	A reference to the left child. This reference might be `null` if the node is a leaf (the node does not have children).
middle child	A reference to the middle child. This reference might be `null` if the node is a leaf.
right child	A reference to the right child. This reference might be `null` if the node is a leaf or if it has two children.
max	This is the biggest element of the subtree that descends from this node.
min	This is the smallest element of the subtree that descends from this node.
$H_{i,j}$	Cryptographic value stored by each leaf.
Leaf	This is a boolean that indicates whether the node is a leaf or not. If the node is a leaf, it has the following data in addition to the previous fields: – The *revocation date*. – The *revocation reason*. – A *certificate identifier* that is formed by the serial number, a hash of the DN of the certificate issuer (CA) and a hash of the public-key used by the issuer (CA) to sign the certificate.

Figure 4 depicts a sample 2–3 tree that represents a set of revoked certificates $\Phi = \{2, 5, 7, 8, 12, 16, 19\}$.

Note that an internal node has only two or three children. If it has two children, these are the "left" and "middle" ones, and if it has three children these are the "left", "middle" and "right" ones. In other words, an internal node always has "left" and "middle" children. A leaf has no children and $min = max = c_j$. Leaves are ordered in the following way: leaves on the left have smaller numbers than leaves on the right.

As mentioned in Sect. 2, apart from the data that identifies the certificate that has been revoked, revocation systems provide the reason and the date of revocation. We compute the following cryptographic value for each leaf to include the previous information in the MHT:

$$H_{0,j} = h\{CertID \,|\, Reason|\ Date\} \tag{5}$$

As pointed out in Sect. 4, the response varies depending on whether the requested certificate belongs to the MHT or not.

If $c_{target} \in \Phi$, the user needs to be provided with the $\mathcal{P}$ath from the target leaf to the root. For this, a recursive algorithm is provided that starts from the root and goes across the tree until the target leaf is reached. During this trip through the tree, the algorithm finds the $\mathcal{P}$ath for the target leaf.

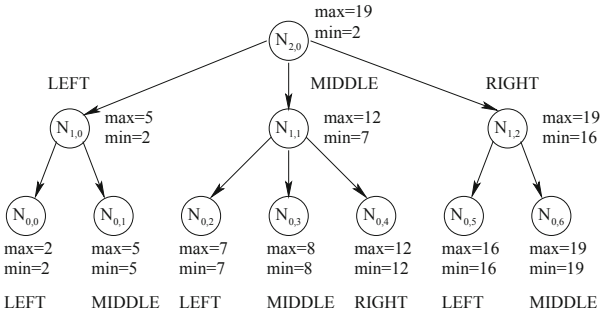

Fig. 4. A sample 2–3 tree

To sum up, when the algorithm has reached a certain internal node denoted by N_i, it decides the next node to go to (denoted by N_{i-1}) and adds the siblings of N_{i-1} to the $\mathcal{P}$ath. The algorithm is presented below in a pseudo-code.

```
While (N_i ≠ leaf){
  If (N_i has two children){
    If (c_target < N_i.middle.min){
      N_{i-1} = N_i.left
      #N_i.middle is included in Path
      N_i.middle ≫Path
      }
    Else {
      N_{i-1} = N_i.middle
      N_i.left ≫Path
      }
  }
  If (N_i has three children){
    If (c_target < N_i.middle.min){
      N_{i-1} = N_i.left
      N_i.middle ≫Path
      N_i.right ≫Path
      }
    Else if (c_target < N_i.right.min){
      N_{i-1} = N_i.middle
      N_i.left ≫Path
      N_i.right ≫Path
      }
    Else {
      N_{i-1} = N_i.right
      N_i.left ≫Path
      N_i.middle ≫Path
      }
```

The above algorithm is illustrated by an example in Fig. 5:

1. Start from the root (Fig. 5# $root = N_{2,0}$, $c_{target} = 16$).
2. Choose next node (Fig. 5# $N_{1,2}$).
3. Add siblings to $\mathcal{P}$ath (Fig. 5# $\{N_{1,0}, N_{1,1}\} \gg \mathcal{P}$ath).
4. Choose next node (Fig. 5# $N_{0,5}$).
5. Add siblings to $\mathcal{P}$ath (Fig. 5# $N_{0,6} \gg \mathcal{P}$ath).
6. End since the target leaf has been reached (Fig. 5# $c_{target} = 16$).

If $c_{target} \notin \Phi$, the two adjacent leaves to the target certificate must be found. Note that if $c_{target} \notin \Phi$ and the same algorithm previously described is followed,

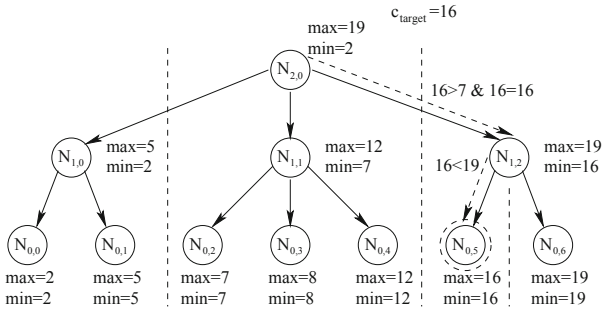

Fig. 5. Example: searching for a revoked certificate

the $\mathcal{P}$ath of the minor adjacent of c_{target} will be found. To find the major adjacent a similar algorithm is needed but using other border-lines. This is why the "Node" object includes the *max* parameter.

5.3 Adding Revoked Certificates

When a certificate has been revoked, it must be inserted into the MHT. The algorithm that proposed for inserting a revoked certificate in the MHT is depicted below and it is also illustrated by an example in Fig. 6.

1. Start searching the target leaf (Fig. 6a $c_{target} = 9$).
2. Stop searching at level 1. The node at which the search algorithm stops is denoted as $N_{1,j}$ (Fig. 6 $N_{1,j} = N_{1,1}$).
3. If $N_{1,j}$ has "2" children, then
   (a) Insert c_{target} as a child of $N_{1,j}$ in the correct position.
   (b) Update $N_{1,j}.max$, $N_{1,j}.min$ and $H_{1,j}$.
   (c) Recalculate the $H_{i,j}$ from the current leaf to the root (note that the resulting tree is balanced).
   (d) End.
4. If $N_{1,j}$ has "3" children, c_{target} would be the fourth child, which is not possible in a 2–3 tree by definition, then
   (a) Split $N_{1,j}$ (a new node is created). The new node is denoted by $N_{1,j+1}$ (Fig. 6b $N_{1,j+1} = N_{1,2}$).
   (b) The two leaves with the smaller serial number remain as children of $N_{1,j}$, while the other two leaves become children of $N_{1,j+1}$.
   (c) Update $N_{1,j}.max$, $N_{1,j}.min$, $H_{1,j}$, $N_{1,j+1}.max$, $N_{1,j+1}.min$ and $H_{1,j+1}$. The father of $N_{1,j}$ is denoted by $N_{2,k}$ (Fig. 6b $N_{2,k} = N_{2,0}$).
5. Apply the algorithm recursively to insert $N_{1,j+1}$ as a child of $N_{2,k}$.

In the last instance, the root node may be split. In this case, a new root is created whose children will be the old root and the new node. The root splitting is how the tree grows (Fig. 6c).

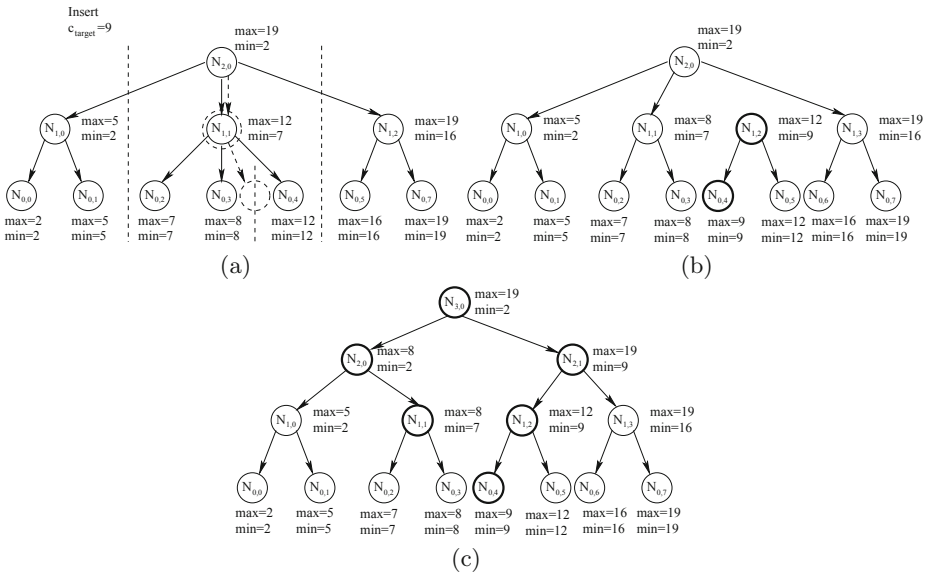

Fig. 6. Example: inserting a revoked certificate (with root splitting)

5.4 Response Verification

To verify a response the user must check that each `TreePath` included in the response is correct to verify a response. In addition, if the target certificate has not been revoked, the user also needs to ensure that the `TreePath`s provided belong to real adjacent nodes.

In first place, to check that each `TreePath` included in the response is correct the user must verify that the `rootHash` computed from the $\mathcal{P}$ath matches the `rootHash` included in the $\mathcal{D}$igest.

However, if a target certificate has not been revoked, this is not enough. The user also needs to ensure that the `TreePath`s provided belong to real adjacent nodes (remember that the repository is a non-TTP, so the user can be misled into believing that a certain pair of nodes within the tree are adjacent leaves).

For example, let's suppose that a user wants to perform a transaction using a given certificate. The certificate is identified by c_{target}. Using the example in Fig. 4, let's assume that $c_{target} = 16$. Note that $c_{target} \in \Phi$, but let's suppose that a malicious repository provides us with the $\mathcal{P}$ath for a couple of leaves that belong to the MHT, claiming that they are adjacent. For instance, let us assume that these leaves are $c_{minor} = 8$ and $c_{major} = 19$. If we only check that $\{c_{minor}, c_{major}\} \in \Phi$, we will think that c_{target} is valid and we will perform the fraudulent transaction.

Thus, an algorithm to check that two nodes are adjacent is also necessary to verify a response. Next, a recursive algorithm is proposed, which verifies, for a given couple of `TreePath`s that they actually belong to adjacent leaves.

The algorithm works without adding any extra information to the data structures. The alleged adjacent leaves are denoted by $N_{0,j}$ and $N_{0,j+1}$. The algorithm for adjacency checking is the following:

1. The user computes $H_{1,m}$ and $H_{1,n}$, which denote respectively the cryptographic values of the fathers of $N_{0,j}$ and $N_{0,j+1}$.
2. If $H_{1,m} = H_{1,n}$, then both leaves have the same father. Then
   (a) If $N_{0,j} = N_{1,m}.left$ and $N_{0,j+1} = N_{1,m}.middle$, then **they are adjacent nodes**.
   (b) If $N_{0,j} = N_{1,m}.middle$ and $N_{0,j+1} = N_{1,m}.right$, then **they are adjacent nodes**.
   (c) Else, **they are not adjacent nodes**.
3 If $H_{1,m} \neq H_{1,n}$, then the leaves do not have the same father. Then
   (a) If $N_{1,m}$ has "2" children and $N_{0,j} \neq N_{1,m}.middle$, then **they are not adjacent nodes**.
   (b) If $N_{1,m}$ has "3" children and $N_{0,j} \neq N_{1,m}.right$, then **they are not adjacent nodes**.
   (c) If $N_{0,j+1} \neq N_{1,n}.left$, then **they are not adjacent nodes**.
   (d) Otherwise, the user computes $H_{2,p}$ and $H_{2,q}$, which denote respectively the cryptographic values of the fathers of $N_{1,m}$ and $N_{1,n}$, and applies the algorithm recursively. In the last instance, the root is the unique common father between the pair of nodes.

Illustrative examples of this algorithm are depicted in Fig. 7.

It must be pointed out that the strength of the above algorithm resides in the position that a certain node occupies relative to its father, in other words whether a certain node is LEFT, MIDDLE or RIGHT. Note that the end user can trust this information since the relative node positions cannot be swapped by a malicious repository because a non-commutative hash function has been used. If the malicious repository modifies the concatenation order, then it will change the cryptographic value of the next step:

$$H_{i+1,k} = h(H_{i,j} | H_{i,j+1}) \neq h(H_{i,j+1} | H_{i,j}) \tag{6}$$

Finally, note that in some cases minor adjacent, major adjacent or both can be missing. For instance,

- If $\Phi = \{\emptyset\}$, i.e. the MHT is empty, then both adjacent nodes are missing.
- If $c_{target} < c_j \ \forall j$, i.e. the serial number of the target certificate is smaller than the smallest leaf within the MHT, then there is no minor adjacent.
- If $c_{target} > c_j \ \forall j$, i.e. the serial number of the target is bigger than the biggest leaf within the MHT, then there is no major adjacent.

A serial number is nothing more than an array of bits. The serial numbers with all bits set to 0 s and 1 s are reserved (not assigned to "real" certificates) to bound the MHT. These "special" serial numbers represent 0 and $+\infty$ respectively, so now each possible serial number has two adjacent nodes independently of the certificates contained by the MHT.

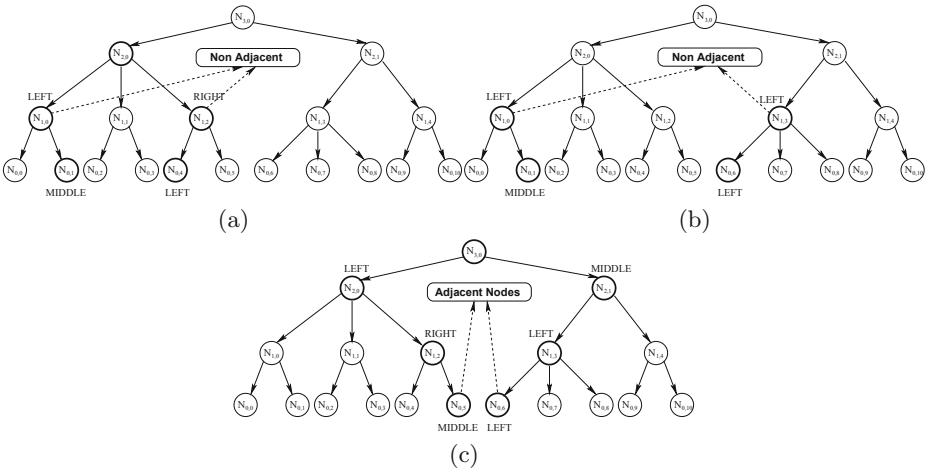

Fig. 7. Examples of adjacent node checking

6 Security Discussions

For a revocation system implementing the mechanism proposed in this chapter to be effective, certificate-using applications must connect to any of the repositories available. In the event that such a connection cannot be obtained, certificate-using applications could implement other processing logic (CRL, OCSP etc.) as a fall-back option.

Another important aspect that the MHT administrators must take into account when deploying the system is that there can be problems with firewalls if the transport mechanism is different from HTTP (many firewalls do not allow anything but HTTP to pass through). In addition, the administrators of the certificate status checking system should not forget that the HTTP transport makes it possible for firewall administrators to configure them to selectively block out messages using specific Multipurpose Internet Mail Extensions (MIME) types. Administrators should also take the reliance of HTTP caching into account because it may give unexpected results if the MHT requests or responses are cached by intermediate servers and these servers are incorrectly configured or are known to have cache management faults. Therefore, deployments should take the reliability of HTTP cache mechanisms into account when MHT over HTTP is used.

On the other hand, possible attacks on the certificate revocation system and their countermeasures must be considered, including:

– *RDI Masquerade Attack:* An attacker or a malicious repository could attempt to masquerade a trustworthy revocation data issuer.
Countermeasures: This attack is avoidable if the user verifies the signature included in the $\mathcal{D}$igest using the correct certificate of the revocation data issuer.

- *Response Integrity Attack:* An attacker or a malicious repository could modify part or the whole of a response sent by legitimate repository.
  Countermeasures: This attack cannot be successfully carried out if the response is verified according to the procedure described in Sect. 5.4. Note that the inherent structure of the MHT together with the response verification algorithm make infeasible to alter an MHT response without making it invalid: the MHT cannot be modified without modifying the root which is signed, and fake adjacent nodes are detected by the algorithm presented in Sect. 5.4.
- *Replay Attack:* An attacker or a malicious repository could resend an old (good) response prior to its expiration date but after the $\mathcal{D}$igest has changed.
  Countermeasures: Decreasing the validity periods of the responses will decrease the window of vulnerability.
- *Denial of Service (DoS) Attack:* An attacker could intercept the responses from a legitimate repository and delete them or the attacker could delay the responses by, for example, deliberately flooding the network, thereby introducing large transmission delays. Note that requests do not contain the repository they are directed to, which allows an attacker to replay a request to any number of repositories. Finally, unsigned error responses open up the algorithm to another DoS attack, in which the attacker sends false error responses.
  Countermeasures: The only way to prevent this attack is to increase the redundancy of repositories, which is easy to deploy since repositories are non-TTPs.

7 Conclusions

The certificate revocation service is critical for the efficient authentication in Vehicular Ad Hoc Networks (VANETs). Decentralized approaches based on reputation 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 VANET restrain their utilization in real-life scenarios. The IEEE 1609.2 standard suggests the use of Certificate Revocation Lists (CRLs) to manage the revocation data. In this context, the problem is that the traditional way of issuing CRLs does not fit well in a VANET where a huge number of nodes are involved and where several pseudonym certificates and identity certificates are assigned to the same vehicle. This chapter has presented the certificate revocation paradigm and reviewed the main revocation mechanisms proposed in the literature.

A novel certificate revocation mechanism based on the Merkle Hash Tree (MHT) has been then presented and discussed. The mechanism introduces an extension to the CRL allowing any non-trusted third party to act as a repository. The main advantage of this extended-CRL is that the road-side units and vehicles can build an efficient structure based on an authenticated hash tree to respond to certificate status checking requests inside the VANET, thus

saving time and bandwidth. Main procedures involved in the proposed mechanism have been described in detail, such as responding to a certificate status request, revoking a certificate, deleting an expired certificate, and response verification. As explained, the proposed certificate revocation mechanism is resistant against malicious behaviors such as Revocation Data Issuer (RDI) masquerading, response modification, replay attacks, and Denial of Service (DoS).

References

1. Bera, R.., Bera, J., Sil, S., Dogra, S., Sinha, N.B., Mondal, D.: Dedicated short range communications (DSRC) for intelligent transport system. In: 2006 IFIP International Conference on Wireless and Optical Communications Networks, pp. 5 (2006)
2. Jiang, D., Delgrossi, L.: IEEE 802.11p: towards an international standard for wireless access in vehicular environments. In: 2008 Vehicular Technology Conference, VTC Spring 2008. IEEE, pp. 2036–2040, May 2008
3. 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, pp. 11–21 (2005)
4. Hubaux, J.P., Capkun, S., Luo, J.: The security and privacy of smart vehicles. IEEE Secur. Priv. **2**(3), 49–55 (2004)
5. Papadimitratos, P., Buttyan, L., Hubaux, J.-P., Kargl, F., Kung, A., Raya, M.: Architecture for secure and private vehicular communications. In: 2007 7th International Conference on ITS Telecommunications, ITST '07, pp. 1–6, June 2007
6. IEEE. IEEE trial-use standard for wireless access in vehicular environments - security services for applications and management messages. IEEE Std 1609.2-2006, pp. 1–117 (2006)
7. Ganchev, I., O'Droma, M.: New personal IPv6 address scheme and universal CIM card for UCWW. In: Proceedings of the 7th International Conference on Intelligent Transport Systems Telecommunications (ITST 2007), pp. 381–386, June 2007
8. Haas, J.J., Hu, Y.-C., Laberteaux, K.P.: Efficient certificate revocation list organization and distribution. IEEE J. Sel. Areas Commun. **29**(3), 595–604 (2011)
9. Wasef, A., Shen, X.: Maac: message authentication acceleration protocol for vehicular ad hoc networks. In: 2009 Global Telecommunications Conference, GLOBECOM 2009. IEEE, pp. 1–6, 30 November 2009–4 December 2009
10. Papadimitratos, P., Buttyan, L., Holczer, T., Schoch, E., Freudiger, J., Raya, M., Ma, Z., Kargl, F., Kung, A., Hubaux, J.-P.: Secure vehicular communication systems: design and architecture. IEEE Commun. Mag. **46**(11), 100–109 (2008)
11. Laberteaux, K.P., Haas, J.J., Hu, Y.-C.: Security certificate revocation list distribution for vanet. In: Proceedings of the 5th ACM International Workshop on VehiculAr Inter-NETworking, VANET '08, pp. 88–89 (2008)
12. Munoz, J.L., Forné, J., Esparza, O., Soriano, M.: Certificate revocation system implementation based on the Merkle hash tree. Int. J. Inf. Secur. (IJIS) **2**(2), 110–124 (2004)
13. Forné, J., Muñoz, J.L., Rey, M., Esparza, O.: Efficient certificate revocation system implementation: Huffman Merkle hash tree (huffmht). In: V Jornadas de Ingeniería Telemática, 09 (2005)
14. Wohlmacher, P.: Digital certificates: a survey of revocation methods. In: 2000 ACM Workshops on Multimedia, pp. 111–114. ACM Press, March 2000

15. 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, pp. 86–87 (2008)
16. Wasef, A., Jiang, Y., Shen, X.: DCS: an efficient distributed-certificate-service scheme for vehicular networks. IEEE Trans. Veh. Technol. **59**(2), 533–549 (2010)
17. 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, pp. 82:1–82:7 (2008)
18. 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)
19. Marias, G.F., Papapanagiotou, K., Georgiadis, P.: ADOPT. a distributed ocsp for trust establishment in manets. In: 2005 11th European Wireless Conference (2005)
20. Moore, T., Clulow, J., Nagaraja, S., Anderson, R.: New strategies for revocation in ad-hoc networks. In: Stajano, F., Meadows, C., Capkun, S., Moore, T. (eds.) ESAS 2007. LNCS, vol. 4572, pp. 232–246. Springer, Heidelberg (2007)
21. Wasef, A., Shen, X.: EDR: efficient decentralized revocation protocol for vehicular ad hoc networks. IEEE Trans. Veh. Technol. **58**(9), 5214–5224 (2009)
22. Raya, M., Manshaei, M.H., Félegyhazi, M., Hubaux, J.-P.: Revocation games in ephemeral networks. In: Proceedings of the 15th ACM Conference on Computer and Communications Security, CCS '08, pp. 199–210 (2008)
23. Wasef, A., Shen, X.: EMAP expedite message authentication protocol for vehicular ad hoc networks. IEEE Trans. Mob. Comput. **12**, 78–89 (2013)
24. Gañán, C., Muñoz, J.L., Esparza, O., Mata, J., Hernández-Serrano, J., Alins, J.: Coach: collaborative certificate status checking mechanism for vanets. J. Netw. Comput. Appl. (2012)
25. Gañán, C., Muñoz, J.L., Esparza, O., Mata-Díaz, J., Alins, J.: Pprem: privacy preserving revocation mechanism for vehicular ad hoc networks. Comput. Stand. Inter. **36**(3), 513–523 (2014)
26. Gañán, C., Muñoz, J.L., Esparza, O., Loo, J., Mata-Díaz, J., Alins, J.: BECSI: bandwidth efficient certificate status information distribution mechanism for VANETs. Mob. Inf. Syst. **9**(4), 347–370 (2013)
27. Gañán, C., Muñoz, J.L., Esparza, O., Mata-Díaz, J., Alins, J.: Epa: an efficient and privacy-aware revocation mechanism for vehicular ad hoc networks. Pervasive and Mobile Computing (2014, in press)
28. Gañán, C., Muñoz, J.L., Esparza, O., Mata-Díaz, J., Alins, J.: Toward revocation data handling efficiency in VANETs. In: Vinel, A., Mehmood, R., Berbineau, M., Garcia, C.R., Huang, C.-M., Chilamkurti, N. (eds.) Nets4Trains 2012 and Nets4Cars 2012. LNCS, vol. 7266, pp. 80–90. Springer, Heidelberg (2012)
29. Merkle, R.C.: A Certified Digital Signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)