Design and Analysis of a Lightweight Certificate Revocation Mechanism for VANET

Jason J. Haas
University of Illinois at Urbana-Champaign
Urbana, Illinois, U.S.A.
jjhaas2@illinois.edu

Yih-Chun Hu
University of Illinois at Urbana-Champaign
Urbana, Illinois, U.S.A.
yihchun@illinois.edu

Kenneth P. Laberteaux
Toyota Research Institute
Ann Arbor, Michigan, U.S.A.
klaberte@acm.org

ABSTRACT

In this paper, we propose a lightweight mechanism for revoking security certificates appropriate for the limited bandwidth and hardware cost constraints of a VANET. A *Certificate Authority* (CA) issues certificates to trusted nodes, i.e., vehicles. If the CA looses trust in a node (e.g., due to evidence of malfunction or malicious behavior), the CA must promptly revoke the certificates of the distrusted node. To distribute revocation information quickly even during incremental deployment, we propose CAs use Certificate Revocation Lists (CRLs). The CRL should be composed in a secure manner, and it should be exchanged in a way such that the CRL is both quickly and widely distributed. Laberteaux et al. [1] proposed a mechanism for the quick distribution of CRL updates that also covers a wide area by using car-to-car (C2C) communication. However, this revocation process, which involves both the CA and VANET nodes, must conform to the aforementioned bandwidth and hardware restrictions. In this paper, we present mechanisms that achieve the goals of reduced CRL size, a computationally efficient mechanism for determining if a certificate is on the CRL, and a lightweight mechanism for exchanging CRL updates. Additionally, we present a formal proof of the security of our mechanism for reducing the size of CRLs.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protections, (e.g., firewalls)*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Design, Performance, Security

Keywords

VANET, security, revocation, CRL

1. INTRODUCTION

In a VANET, vehicles will rely on the integrity of received data for deciding when to present alerts to drivers. Further in the future, this data may be used as the basis of control decisions for autonomous vehicles. If this information is corrupted, vehicles may present unnecessary or erroneous warnings to their drivers, and the results of control decisions based on this information could be even more disastrous. Information can be corrupted by two different mechanisms: malice and malfunction. Similarly, vehicles have two defense mechanisms: an internal filter and external reputation information. The former defense mechanism can consist of filters based on physical laws (e.g., maximum braking deceleration, maximum speed, physical space constraints) [2]. The latter defense mechanism can consist of reports from other vehicles or entities on the validity or trustworthiness of data originating from certain vehicles. In this paper, we will concern ourselves with the latter defense mechanism.

Information received from corrupted nodes should be disregarded or not trusted by legitimate vehicles, otherwise, a malicious vehicle could, for example, obtain a less congested route for itself by overstating the number of vehicles on its desired roadway. As a second example, a corrupted node could trigger erroneous driver warnings to be displayed in other vehicles by falsifying its position information. IEEE 1609.2, the trial-use standard concerning security services for vehicular environments, stipulates that vehicles will be authenticated using certificates issued by a *Certificate Authority* (CA) in a *Public Key Infrastructure* (PKI) setup [3]. Illegitimate vehicles should have these certificates revoked, and the identity of the revoked certificates (although ideally not the identity of the associated driver) should be published and distributed to legitimate vehicles. Whatever mechanism that is used for distributing this revocation information should distribute the information securely, quickly, and broadly in order to limit the amount of damage illegitimate vehicles can do.

In a VANET, the CA can utilize Road-Side Units (RSUs) to distribute this revocation information. However, the density of RSU placement (i.e., how many RSUs are set up in a certain area) has not yet been determined. In fact, it is likely that RSUs will be sparsely placed, and thus, vehicles may spend significant time outside radio contact of an RSU [4]. Even if RSUs are eventually deployed with sufficient density, a VANET must be able to operate during stages of incremental deployment, that is, before a sufficient density of RSUs come online.

In previous work, Laberteaux et al., proposed that revocation information be distributed in the form of a Certificate Revocation List (CRL) in an epidemic manner, that is, through car-to-car (C2C) communication [1]. This epidemic distribution mechanism supplements RSU-based distribution[1], essentially turning vehicles into mobile distributors, making use of vehicles' mobility. The use of C2C distribution of CRLs addresses incremental deployment issues. The authors evaluated the performance of their epidemic distribution mechanism through simulation and have shown that their mechanism provides significant advantages over a RSU-only CRL distribution mechanism in terms of speed and breadth of network coverage.

Summary.

In order to reduce the potential network and computational overhead imposed by any CRL distribution mechanism, we propose optimizations for organizing, storing, and exchanging CRL information. Our first optimization is that certificate identifiers for a single node be generated using a secret key s and a block cipher (e.g., AES) keyed by s or a secure Pseudo-Random Number Generator (PRNG) where s is used as the seed. Either of these mechanisms allows the CA to revoke a vehicle's certificates with a single addition to the CRL (i.e., s), thus minimizing the size of the CRL or CRL updates. We present a mechanism for a single CA to revoke a vehicle in an efficient manner. We prove that our optimizations result in no more privacy lost than the privacy lost simply due to broadcasting a certificate. In Appendix A, we extend our scheme to allow for two CAs, neither of which holds the complete key s, rather both combine information they hold about s only during revocation, thus providing unrevoked vehicles with privacy from a single CA.

Second, we propose that certificate identifiers be stored in a Bloom filter, which is a probabilistic data structure (i.e., searching has a non-zero, but ideally small, false positive rate) and has a constant ($O(1)$) cost in terms of computation for searching and storage. This approach was, to our knowledge, first proposed by Raya et al. [5], who analyzed Bloom filter false positive rates, and suggested that Bloom filters can be used for storing CRLs. We addresses the important question of what computational resources are required for a Bloom filter implementation. This data shows that using a Bloom filter requires a small amount of computational overhead when it is used to search for certificates in the CRL. We also compare using Bloom filters with other deterministic data structures. Finally, we discuss the effects of a Bloom filter's false positive rate and how to alleviate those effects.

Our final optimization regards how CRL information should be exchanged in the network so that network bandwidth is conserved. We propose that CRL updates, rather than the whole CRL, be exchanged, similar to delta CRLs [6]. Each time a CRL is passed to a receiver, the sender sends only the revocation information that the receiver does not have, and includes a signature (generated by the CA) that covers the entire CRL (including the parts not sent). For example, if there are 1000 entries in the CRL, but the receiver already has 997 of them, then the sender need only send 3 entries and one signature, substantially reducing the overhead of CRL transfer.

We motivate our research in Section 2. In Section 3, we present our organizational optimization for certificates and prove its security. We analyze the performance of our mechanism for certificate storage in Section 4. We propose efficient mechanisms for exchanging CRL information in Section 5. In Section 6, we compare our proposals with previous work. We conclude in Section 7. In Appendix A, we present our multi-CA mechanism for certificate organization and revocation.

[1]The CA may utilize other communication methods to distribute CRLs, such as cellular or WiFi. However, the combining of multiple communication technologies within a vehicle's safety computer is not yet a certainty. Further, if a subset of VANET nodes can receive a CRL via an alternative communication technology, the epidemic distribution method can be used to propagate the CRL to other VANET nodes that lack this alternative channel.

2. MOTIVATION

Many governments, automotive companies, and government and industry consortia[2] have made the reduction of vehicular fatalities a top priority [7; 8], due to the immense loss of life and the resulting economic impact vehicular collisions have. On average, vehicular collisions cause 116 deaths and 7900 injuries daily in the United States, leaving an economic impact of \$230 billion [7; 9]. The damage is similarly devastating in the European Union, where there are more than 100 deaths and 4600 injuries daily, costing €160 billion annually [10].

The U.S. National Highway Traffic Safety Administration itself and these consortia have identified Dedicated Short Range Communications (DSRC) [11; 12; 13] as the leading wireless technology for improving vehicular safety [14]. DSRC also supports other applications, such as electronic toll collection, traffic congestion probing, and other infotainment and e-commerce applications. Significant progress has been made in standardizing the lower layer protocols for DSRC. Further, several industry/government consortiums strive to identify which vehicular applications (and related technologies) will provide the greatest benefits.

These consortia have identified security and privacy as important issues for DSRC deployment. As mentioned above, abuses, intentional or otherwise, of the DSRC system could threaten the safety of vehicles. Additionally, laws and user acceptance require that users of DSRC maintain as much privacy as possible. (The various consortia have formally defined privacy in a variety of ways [15; 16].) Simultaneously meeting both privacy and security goals shrinks the design space considerably.

3. CERTIFICATE ORGANIZATION

For nodes to communicate securely, they must possess a copy of each other's credentials in the form of a certificate. Nodes can exchange certificates upon first encounter. An example is shown in Table 1. Certificates can be pre-installed or updated during production and periodic service appointments, which may potentially reduce the necessity of certificate exchanges.

In a VANET, there is a tension between privacy and authentication. Received safety messages must be trusted, and this requires trusting the sender. However, senders wish to retain as much privacy as possible, which implies a receiver must trust, but not conclusively identify, the sender.

Clearly, if a vehicle V has only one certificate, even if it is identified by a pseudonym certificate ID, P, then tracking the vehicle via passive recording of DSRC messages is possible. An attacker,

[2]e.g., the Crash Avoidance Metrics Partnership (CAMP), which includes the NHTSA, Mercedes, Ford, GM, Honda and Toyota (see `http://www-nrd.nhtsa.dot.gov/departments/nrd-12/pubs\_rev.html`), the Cooperative Intersection Collision Avoidance (CICAS) consortium (see `http://www.its.dot.gov/cicas/index.htm`), the Vehicle Infrastructure Integration Consortium (VII-C) (see `http://www.vehicle-infrastructure.org/`), the Car2Car Communications Consortium (see `http://www.car-2-car.org/`) and the related Sevecom project (see `http://www.sevecom.org`), and the Advanced Safety Vehicle (ASV) Project, which is a partnership of Japan's 14 automobile, truck, and motorcycle manufacturers, sponsored by Japanese Ministry of Land, Infrastructure and Transport (`http://www.mlit.go.jp`)

Table 1: An example of a generalized public key certificate exchange. When vehicle A hears a message from a previously unknown vehicle X, it replies with a query for the certificate and short-term public key of X.

X→*: $\langle Message_i \rangle$	A receives a broadcast from X but does not have X's short-term key.
A→*: $\langle cert_A, pub_A, X \rangle$	A broadcasts its certificate, its signed short-term public key, and a request for X's certificate.
X→*: $\langle cert_X, pub_X \rangle$	X broadcasts its certificate and its signed short-term public key.

A, who attacks privacy needs only to link P to V one time (e.g. by listening to messages sent by V near V's home location). Then, by employing listening stations near a location of interest, A can determine if and when V visits that location.

3.1 Certificate Groups

To make it more difficult for A to track V, V could be issued a group of certificates, each with a different ID. V would then use different certificates at different times and locations. Of course, if A is willing to follow V to collect all (or even many) of V's certificate IDs, multiple certificates will not improve V's privacy. However, V has little hope of retaining location privacy from such a determined attacker, that is, if A tracks V, A no longer needs DSRC certificates to track V's movements. (For example, a determined attacker can easily track vehicles using license plates.)

Further, issuing V a set of certificate IDs that are easy to relate does little to improve V's privacy. For example, issuing V the block of IDs C4ED34ACBFF80000 ... C4ED34ACBFF8FFFF would make V easy for A to track. Instead, the certificate IDs issued to V should be very difficult for most entities to associate. From the perspective of V's privacy, it is best if V's certificate IDs appear to be completely random.

To make use of the group of certificates it is issued, V must change the certificate it is currently using. We define a *certificate transition* as the event when vehicle V switches from one certificate to another. It would be relatively easy for those entities within communication range of V at the time of a certificate transition to link these two certificates to the same vehicle. This is made even easier if the messages include information about V's current and future location (as some safety applications require [16]). Observing a large number of V's certificate transitions would allow the creation of a certificate profile for V, even if the true identity of V is not known.

It is useful to consider how many certificates V might be issued. This number should be large enough so as to make observing a large number of certificate transitions difficult for all but the most determined attacker. This concern is removed if certificates are used only once for a period of time and then never reused. However, the number of certificates issued to a vehicle is limited by storage constraints. If certificates are used only once, then there must be reliable opportunities for the vehicle to replenish its list of certificates, or vehicles must be preloaded with a sufficient number of certificates for the expected vehicle lifetime. On the question of the size of the certificate group, we do not offer definitive recommendations, but instead describe the logic that influences our designs.

3.2 Size of Certificate Revocation Lists

A recent study found that a typical American spent 15 hours/week in a car [17]. As this included time spent as a driver or as a passenger, we reason that an average US vehicle operates no more than 15 hours/week. If each certificate is used for a single ten-minute period and never reused, then 4660 certificates are consumed each year. Therefore, a vehicle will need approximately 5000 certificates per year (fewer, if they are reused). If an opportunity to replenish a vehicle's certificates occurs at least once every five years, a vehicle must be able to store approximately 25,000 certificates. If each certificate is approximately 100 bytes, this requires a storage size of approximately 2.5 MB, which is on the order of on-chip FLASH memory on currently available automotive embedded processors[3].

Thus, it is possible that a vehicle will possess 25,000 certificates at one time. It is desirable that these certificate IDs be difficult to relate. A CA may need to revoke all certificates held by a given vehicle. If the certificate IDs for the revoked vehicle are truly random, i.e., there is no compact representation for the certificate group, then the CA would need to explicitly identify all 25,000 certificates of the revoked vehicle in the updated CRL. Assuming that certificates can be identified by a 16 byte fingerprint (the size of one AES block), to revoke a single vehicle, the CRL would thus grow by 400 kB. Since the CRL must be widely propagated, such growth in the size of the CRL would result in excessive computational and storage overhead, and network bandwidth consumption.

Algorithm 1 Certificate generation algorithm.

```
Require: $M \equiv$ number of certificates for this vehicle
  Begin: certificate loading for vehicle $V$
  $s \leftarrow generate\_revocation\_key()$
  for $r = 1$ to $M$ do
    $({K_r}^{+}, {K_r}^{-}) \leftarrow generate\_public\_private\_key()$
    $sig_{CA,r} \leftarrow sign(H(\{E_s(r), {K_r}^{+}\}), {K_{CA}}^{-})$
    $Cert_r \leftarrow \{E_s(r), {K_r}^{+}, sig_{CA,r}\}$
  end for
  $send(V, \{Cert_1, \ldots, Cert_M\})$
  End
```

3.3 Privacy-Preserving Certificate Groups

We now present our method for creating a group of certificate IDs for one vehicle V that appear to be unrelated, but in fact are related by a single secret value s known only to the CA. Only for the certificate loading phase of vehicle operation will the CA be required to be online, which will likely occur in a centralized location (e.g., an automobile factory). The method we propose in this section is presented in pseudo-code as Algorithm 1.

To obtain this feature, certificates must include a new field containing a single integer. First, the CA chooses a secret revocation key s for each vehicle. This revocation key s is the key to a *block cipher*, which takes fixed length inputs and permutes them into outputs of the same length, where the particular permutation chosen is based on the key s. Alternatively, s can be the seed for a PRNG rather than the key to a block cipher. Next, the CA creates M public/private key-pairs (above, we argued that M=25,000). $(K_r^+, K_r^-), r \in [1, M]$. Then the CA creates a corresponding group of M certificates. Then for each r it creates a certificate,

$$Cert_r = \{E_s(r), K_r^+, sig_{CA}\}$$

where $E_s(r)$, defined as the certificate ID and is the value produced by encrypting the integer r using the block cipher with the symmetric[4] key s (or for a PRNG the r-th output of the PRNG with seed

[3]The Freescale MPC5554, as advertised in April 2008, has 2 MB of FLASH memory.

[4]As we will show below, any strong cipher will work, e.g. AES-128

s) and sig_{CA} is the CA's digital signature over $\{E_s(r), K_r^+\}$ using the CA's private key. The CA then issues these M key-pairs and associated certificates to vehicle V, either before the vehicle begins its service, or during the replenishing events discussed above.

3.4 Revocation Method

When the CA wishes to revoke vehicle V, it simply appends the value of s to the CRL. We describe CRL delivery methods below. When a vehicle receives a new CRL that includes a new secret s, it encrypts each r, which are know to every vehicle the certificates in its cache using the symmetric key s. If the result matches the certificate ID, this certificate is revoked.

To increase the revocation list management efficiency, a Bloom filter [18] can be loaded with revoked certificate identifiers. To do this, a vehicle generates all possible revoked $E_s(r)$ given a key s. The resulting certificate identifiers can then be included in a Bloom filter. We will discuss what should be done if a certificate identifier results in the Bloom filter indicating a match in Section 4, since our discussion there concerns design parameters for the Bloom filter. Further algorithm extensions employing Bloom filters can be used to efficiently prune cached certificates upon receiving a new revocation key s. We present an analysis of the computational overhead imposed by using Bloom filters in Section 4.1, in light of the discussion of our setup in Section 3.2.

3.5 Privacy properties

The desired privacy property is that it should be difficult for an attacker to relate the certificates belonging to one vehicle (i.e., one certificate group) without knowledge of s. To make this association, the attacker must be able to break the block cipher or PRNG used to generate $E_s(r)$.

Unless the attacker observes vehicle V transition between two certificates, the attacker cannot relate two of V's certificates under practical conditions (i.e., the attacker cannot break the cipher used or track the vehicle with an attached GPS tracking device, etc.). Such an attacker would already have compromised V's location privacy. The attacker, and every other node, will be able to group V's certificates once the revocation key s for V is exposed. This quick association allows for efficient representation of CRLs.

3.6 Discussion

Combining the proposed revocation key approach with the hash function-based nonce selection and Bloom filters for fast certificate acceptance, the certificate generation and revocation scheme has the following properties:

1. Certificates grow by only a small amount for $E_s(r)$ (one block; 16 bytes for AES).

2. All of a vehicle's keys can be represented in the CRL with a small amount of data: the revocation key, s, (of size 16 bytes for AES), and a count of the number of certificates granted for this key (the latter field could be omitted if all cars have the same number of keys).

3. Any additional loss in privacy is due only to imperfections in the block cipher or PRNGand not due to the protocol design itself.

4. With memory usage linear in the number of revoked vehicles, each certificate can be checked in time linear in the number of revoked vehicles (no-precomputation approach).

5. With memory usage linear in the number of revoked keys, most certificates can be checked very quickly using a Bloom filter with few false positives. (Bloom filters use memory linear in the number of elements it is meant to hold, for a fixed false positive rate and a fixed number of index functions.)

6. The network overhead involved in transferring CRL information regarding a single vehicle is constant (independent of the number of certificates that vehicle holds) because only the revocation key, s, of a revoked vehicle and an optional number of keys field, M, is added to the CRL.

4. REVOKED CERTIFICATE STORAGE

As discussed above, we propose that vehicles use Bloom filters to store all revoked certificates. To be clear, the Bloom filter is not populated with values of s received by CRLs. Instead, upon receiving a new value, s, in a CRL, the receiver calculates all possible certificates associated with that s and populates the Bloom filter with these revoked certificate IDs ($E_s(r)$). $E_s(r)$ can be calculated by vehicles holding the CRL since the CRL contains the revocation key s and the range of r (M) (M may not be included if it is constant and therefore implied). It is important to remember that the CRL will contain revocation keys, s, while each vehicle will insert into its Bloom filter certificate IDs, $E_s(r)$. Thus, vehicles can efficiently check both certificates previously received and certificates they receive in the future to see if these certificates have been revoked. In this section, we discuss our results from analyzing the performance for practical Bloom filter implementations. Additionally, we discuss the consequences of using a Bloom filter, which is a probabilistic data structure, that is, when searching a Bloom filter, false positives occur with some probability. We compare our use of Bloom filters with deterministic data structures.

4.1 Bloom Filter Performance Analysis

A Bloom filter is a data structure that has constant computational cost ($O(1)$) for insertion and search operations. However, Bloom filters are a probabilistic data structure, meaning that there is a chance that a search operation may indicate a data item is in the filter when it actually is not. In the context of this paper, a false positive occurs when a non-revoked certificate appears to be a revoked certificate, that is, appears to be in the Bloom filter. We propose an approach to addressing the problem of false positives at the end of this section. A desired false positive rate can be chosen by setting 3 parameters: the number of elements to be inserted (n), the size of the data structure (m, here specified in bits), and the number of hash or index functions used for insertions and searches (k). Using this notation, the false positive rate can be calculated as, [19]

$$P(\text{false positive}) = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k} \tag{4.1}$$

To illustrate, let us assume that 1 in 100,000 vehicles are malicious or malfunctioning in a VANET with 100,000,000 vehicles, then there are 1,000 vehicles whose certificates should appear on the CRL. If each vehicle possesses 25,000 certificates, as mentioned above, then the CRL should contain 25,000,000 certificate identifiers. Thus, n = 25,000,000. If we choose $m = 2^{28}$ bits (32 MB) and $k = 6$, then we obtain a false positive rate of 0.616%, or approximately 1 in every 160 certificates. We believe this false positive rate is acceptable for reasons described below.

To evaluate the computational overhead of using Bloom filters, we implemented a Bloom filter, using $k = 6$ hash functions based on MD-5 for index functions and a vector of $m = 2^{28}$ bits for the Bloom filter storage. The input to our hash functions were randomly generated 128-bit certificate IDs. Our hash functions were

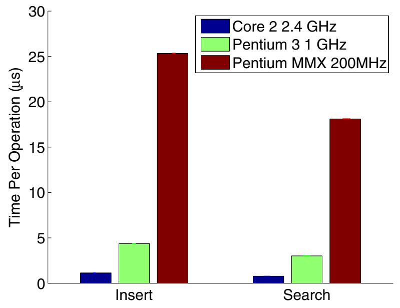

Figure 1: Performance analysis of Bloom filter insertions and searches of vehicle certificate IDs.

Table 2: Intel Core 2 2.4 GHz insert and search times per certificate

Data Structure	Insert (μs)	Search (μs)
Bloom filter	2.6	1.4
AVL tree	6.8	3.4
Red-black tree	5.6	3.6

implemented as follows. First, we concatenated a random byte repeated 16 times, the input certificate ID, and the random byte repeated 16 times again. Next, we repeated this step but used a different random byte. (In a deployed VANET implementation, these two padding bytes would be fixed and be specified in a standard.) We separately hash each of these 2 48-byte strings using MD-5. This concatenation mechanism is derived from previous work by Fan et al. [19]. We used a single MD-5 output as 3 different hash functions, taking the 3 indices from 3 different 4-byte blocks of the output. Essentially, we called MD-5 2 times with 2 different random bytes padding the random certificate IDs, as described previously. Then we used 3 4-byte blocks from each output to generate 28-bit indices for the Bloom filter. By reusing the MD-5 outputs, we were able to double the filter's performance. We used the OpenSSL implementation of MD-5 for our tests [20].

We tested the performance of both inserting certificate IDs into and searching for certificate IDs in the Bloom filter. Our certificate IDs consisted of 16 bytes from a cryptographically strong PRNG, also obtained from OpenSSL [20]. We used a cryptographically strong PRNG to emulate the output of our certificate ID generating scheme, as described above. For each insertion or search test, we inserted or searched for 25,000 certificate IDs, which is the number of certificate IDs a vehicle would possess according to our discussion in Section 3.1. We recorded the time taken for each batch of tests (25,000 insertions or searches) to microsecond precision, and we performed each batch of tests 10,000 times on each test machine. For the search benchmarking, half of the identifiers sought were in the filter and half of them were not in the filter.

We performed our tests on three different machines, each of which has different hardware. The first machine had an Intel Core 2 (2.4 GHz), the second machine had an Intel Pentium 3 (1 GHz), and the third machine had an Intel Pentium MMX (200 MHz). Figure 1 shows the time it takes on each processor to insert or search for a single certificate ID. These performance tests indicate that all of the certificates from a revoked vehicle could be added to a vehicle's internal CRL data structure (Bloom filter) in 10's to 100's of milliseconds. Also, searching for a certificate in the CRL is 1000's of times faster than the cost of verifying the CA's signature of a certificate. Both of these operations, Bloom filter search and CA signature verification, should be performed by vehicles when they receive a new certificate. The time to search is slightly faster than the insertion time because a search indicates that a certificate ID is not found as soon as a single hash function output is found to not be in the Bloom filter, thus not all hash functions will need to be calculated.

4.2 Data-Structures

Deterministic vs. Probabilistic.

Bloom filters are probabilistic data structures, thus, there is a chance of false positives when searching them. Consequently, it may be preferable to use a deterministic data structure, such as a linked list or balanced binary tree, to store revoked certificate IDs so that no false positives occur. Previously in this section, we concluded that there could be up to 25,000,000 certificates revoked under our assumptions. With 16-byte certificate IDs, each vehicle would require 381 MB to store all of the revoked certificate IDs. In our discussion above, we found that a Bloom filter with a false positive rate of 0.616% requires only 32 MB of storage. If we use a balanced binary tree to store the revoked certificate IDs, the complexity of searching for a single certificate ID is $O(\log n)$.

To compare the performance of deterministic versus probabilistic data structures, we implemented inserting and searching for certificates in AVL trees and red-black trees as well. For the each type of tree, we again populated the tree with 25,000,000 certificates, then searched for and inserted (in that order) 25,000 certificates each. We only tested the performance of each tree on our Core 2 machine because it was the only machine with enough memory to run the tests without accessing a disk[5]. The search times for the trees we used were more than 2 times slower than the search time required with the Bloom filter we tested. However, for a processor with equivalent performance to our Core 2 machine, inserting all of the 25,000 certificates of a revoked vehicle could be done in less than 100 ms. Thus, using a deterministic data structure may not have significant additional processing time overhead, but may be unrealistic for immediate deployment because of the memory requirements.

False Positive Rate.

We believe that having a false positive rate is acceptable for a number of reasons. A certificate from a non-revoked vehicle could trigger a false positive when using a Bloom filter, that is, the certificate could appear to be revoked though that certificate may not be revoked. A vehicle can avoid triggering a false positive by eliminating any of its own certificates that will trigger a false positive. Before using or transmitting a specific certificate, a vehicle can check to see if the certificate will generate a false positive, given the established CRL known to the network. If the certificate will create a false positive, the vehicle can just discard it before it is ever sent, thus preventing a receiver from generating a false positive, assuming every vehicle uses standardized hash functions for the Bloom filter. This discarding can be overcome by simply loading a fraction more certificates initially. However, under this approach, it is more important for a vehicle to have an updated CRL, so that the vehicle can know whether a certificate will produce a false positive when it changes certificates.

We now calculate the fraction of additional certificates required to be pre-loaded on vehicles so that vehicles have a desired number (in expectation) of certificates that do not trigger false positives. The terms we use in our analysis here are summarized in Table 3. The desired number of certificates per vehicle is c = 25,000, as

[5]Our AVL and red-black tree tests used GNU libavl on Linux (see `http://www.stanford.edu/~blp/avl/index.html`) and required approximately 2 GB and 1.5 GB of RAM, respectively.

Table 3: Bloom filter pre-loading parameters

c	Number of certificates desired per vehicle
k, n, m	Bloom filter parameters
l	Number of certificates loaded per vehicle
N	Number of vehicles in the network
f	Fraction of revocable vehicles
p	Fraction additional certificates to load
P_{fp}	Probability of a false positive

we used above. Using the same assumptions as we used in the previous section, we set the number of vehicles in the network to be $N = 100{,}000{,}000$. We increase the fraction of revocable vehicles from our assumptions above to $f = 10^{-4}$ in order for the Bloom filter performance to be more robust to having more revocable vehicles than estimated. We denote the additional fraction of certificates that need to be pre-loaded on vehicles as p.

Assuming certificate identifiers are uniformly and independently distributed among vehicles, the expected number of certificates triggering false positives per vehicle is,

$$E[P_{fp}] = P_{fp}(1+p)c = pc \tag{4.2}$$

We set this expected value to be equal to the number of additional certificates to be loaded on each vehicle (pc).

From Fan et al. [19], we use the approximation for k that minimizes the number of certificates that trigger false positives. We denote this optimized number of hash functions as k^*.

$$k^* = \arg\min P_{fp}(k) \approx \frac{m}{n} ln2 \tag{4.3}$$

We calculate n, the number of entries in the Bloom filter as,

$$n = Nfl = Nf(1+p)c$$

Substituting these equations in for k and n in to (4.1) and rearranging, we get,

$$\frac{pc}{(1+p)c} - \left(1 - (1 - \frac{1}{m})^{kn}\right)^k = 0 \tag{4.4}$$

Choosing $m = 2^{31}$, we solved Equation 4.4 numerically to get $p = 1.767\%$, which results in $k^* = 6$. Thus, an additional 442 certificates are required to be pre-loaded on vehicles to replace certificates that are expected to generate false positives. 442 certificates equates to less than 3 days of usable certificates on a schedule of recharging vehicles' certificates every 5 years, using certificate at the rate of 1 every 10 minutes, as we discussed above. If our assumption of $f = 10^{-5}$ is accurate and the parameters (k^* and , m) for $f = 10^{-4}$ are used, the expected number of false positives is negligible.

5. EFFICIENTLY DISSEMINATING CERTIFICATE REVOCATION LISTS

As discussed above, if the CA wants to invalidate a vehicle's certificates, it must append the vehicle's revocation key, s, to the CRL. The CA would then distribute the CRL so that vehicles can identify (and likely distrust) the newly revoked vehicle. The distribution of the complete CRL to every node in the network may consume significant bandwidth. In this section, we present a CRL update mechanism to reduce the bandwidth required by distributing CRLs, which is similar to delta CRLs [6]. We will assume, in this section, that there is a single CRL, either due to there being a single CA for the VANET, or because each vehicle is concerned only with the CRL for the geographical region in which it is currently operating. Border crossings are prime locations for placing RSUs such that crossing vehicles obtain the latest CRL for the region they are entering.

Algorithm 2 Certificate exchange and update algorithm.

```
$my\_CRL \equiv$ the CRL held by this vehicle
$my\_CRL\_version \equiv$ the version of he CRL held by this vehicle
$my\_diff\_version \equiv$ lowest version of CRL
                    advertised by another vehicle
**Begin:** Upon receipt of broadcast message $m$
**if** ($is\_CRL\_update(m)$) **then**
  **if** ($CRL\_version(m) > my\_CRL\_version$) **then**
    $CRL\_diff \leftarrow get\_CRL\_diff(m, my\_CRL)$
    $temp\_CRL \leftarrow concat\_CRL(my\_CRL, CRL\_diff)$
    **if** ($check\_sig(temp\_CRL, get\_sig(m))$) **then**
      $my\_CRL \leftarrow temp\_CRL$
      $my\_CRL\_version \leftarrow CRL\_version(m)$
    **end if**
  **end if**
**else**
  **if** ($CRL\_version(m) < my\_diff\_version$) **then**
    $my\_diff\_version \leftarrow CRL\_version(m)$
  **end if**
**end if**
**Periodically:**
**if** ($my\_diff\_version < my\_CRL\_version$) **then**
  $broadcast(create\_CRL\_update\_msg(my\_CRL,$
              $my\_diff\_version, my\_CRL\_version)$
  $my\_diff\_version \leftarrow my\_CRL\_version$
**end if**
```

Since a CRL update will be rebroadcast many times, the CRL update size should be small. As a result, we propose that only the entries necessary to bring the receiver's CRL up to the latest version be transmitted by the sender to the receiver. The update will also include a CA signature over the entire CRL, which can then be verified by the receiver. Algorithm 2 shows how updates are generated and distributed among vehicles. The following example will illustrate.

Consider the case where the sender has 1000 entries in its CRL (the number of entries is equivalent to the version number in Algorithm 2): $s_1, s_2, \ldots, s_{1000}$[6]. The sender also has the CA's digital signature over all 1000 revocation keys, $sig_{CA}(H(s_1|s_2|\ldots|s_{1000}))$, where $H()$ is a cryptographic hash function and $|$ denotes concatenation. Therefore, the sender's CRL is $[s_1, s_2, \ldots, s_{1000}, sig_{CA}(H(s_1|s_2|\ldots|s_{1000}))]$. Suppose, the receiver has 997 entries in its CRL: $s_1, s_2, \ldots, s_{997}$, so its CRL is $[s_1, s_2, \ldots, s_{997}, sig_{CA}(H(s_1|s_2|\ldots|s_{997}))]$.

The sender advertises that the size of its CRL is 1000. The receiver advertises that its CRL is of size 997. The sender deduces that the receiver needs the last three values of its CRL. It therefore transmits these three revocation keys as well as the signature over *all* entries, $[s_{998}, s_{999}, s_{1000}, sig_{CA}(H(s_1|s_2|\ldots|s_{1000}))]$. The receiver can then check the CA signature by verifying $sig_{CA}(H(s_1|s_2|\ldots|s_{1000}))$. (The receiver already has values $s_1, s_2, \ldots, s_{997}$.) If the signature is verified, the receiver updates its CRL to be identical to the sender's, $[s_1, s_2, \ldots, s_{1000},$

[6]In Section 3, the certificate ID is denoted as $E_s(r)$, and s is the revocation key (16 bytes).

$sig_{CA}(H(s_1|s_2|\dots|s_{1000}))]$, and begins to advertise its CRL size to be 1000.

The above example showed only a single receiver. However, if there are multiple receivers with varying CRL sizes, the sender simply broadcasts the update with the largest number of missing values. Each updated receiver then appends to its CRL the missing values and checks the final signature (over the 1000 certificates in the above example).

In our example, a revocation key is never removed from the CRL. It may be desirable to remove a revocation key from the CRL if, for example, a vehicle's certificates expire. A direct extension of the above can apply to cases where revocation keys are pruned. However, this would require new signatures for the newly pruned CRL to be generated by the CA and propagated. Hybrid solutions can be adopted by including signatures over various numbers of revocation keys in the updated CRL, that is, over blocks of revocation keys in the CRL if the complete CRL is subdivided. For example, the updated CRL could include one signature for the CRL before pruning and a second signature for the CRL after pruning, allowing nodes to decide which signature to check. CRLs would then need to include information as to which signatures they include or which signatures have been pruned out.

In our discussion above, we supposed that of 100,000,000 vehicles, 1 in 100,000 might get revoked (1000 vehicles). The size of the section of the CRL containing revocation keys is thus 16,000 bytes, which can be sent in only a few packets (considering a standard Ethernet MTU of 1500 bytes). Generally, the full CRL would never need to be sent, and much smaller transfers could be used. If a vehicle is missing a prefix of the CRL (i.e. missing revocation keys $s \in [1, m], m < N$, where N is the version of the CRL), then the entire CRL can be sent to such a vehicle since the size of the CRL is small.

When a node receives a new CRL, it must remove any cached certificates identified as revoked. This pruning effort can be done in conjunction with the Bloom filter insertion. In particular, when inserting $E_s(r)$, certificates matching the $E_s(r)$ pattern should also be removed from the vehicle's cache.

6. RELATED WORK

The U.S. Federal Communications Commission has set aside spectrum in the 5.9 GHz range for DSRC [13]. Development of related standards and technology for DSRC are ongoing [11]. Industry and government consortiums view authentication and encryption as necessary components of a DSRC system. Most, if not all, have chosen to use public key cryptography to accomplish these requirements, where key and certificate management follow a *Public Key Infrastructure* (PKI) model. A PKI employs a trusted CA to assign and bind public keys to identities[7] using the *Digital Signature Algorithm*; these bindings are called *certificates*. These cryptographic primitives are discussed at length by Schneier [21]. The CA must be a trusted entity, likely a government agency or its proxy.

Xu [22] and Korkmaz [23] present preliminary ad hoc protocol designs to support acceptable safety broadcast message reception over a single safety 802.11 DCF channel.

Parno and Perrig [24] discuss various security issues in vehicular networks, including some attempts to address privacy. They propose methods for securing VANET applications, and the use of anonymizers for preserving a vehicle's privacy.

[7]In VANET, to enhance privacy, certificate IDs will most likely be pseudonyms.

Hu and Wang [25] propose silent periods between transmissions to improve privacy. This proposal is generally not feasible for VANET safety applications. In both of the two preceding papers, the authors do not consider the issue of revocation.

Raya and Hubaux [26] propose the use of public key ECDSA signatures and show that computation in a vehicular environment is feasible. The authors concur that RSUs may be sparse, especially during incremental deployment.

Jakobsson and Yung [27] propose a mechanism called "Magic Ink", which provides a way of producing blinded signatures that can be revealed by the signer if misbehavior is detected. The signers can be either centralized or distributed. In the latter case, the integrity of the blinded signatures depends on a (t, n)-threshold scheme. Magic ink is based on using DSS to produce signatures. To revoke a signature, the CA can release what the authors call a *signature–view invariant*. This invariant has properties that match it to a specific signature.

Fischer et al., [28] propose a use of magic ink with a two-part CA, which allows the vehicle to remain anonymous during certificate signing. Vehicles can be tracked or revoked by the CA if t out of n of the CAs cooperate (that is, a (t, n) threshold scheme is used). However, anonymity from the CA is lost if the two CA entities share information. With the magic ink scheme, or with any magic ink scheme, revocation must be done by matching the signature-view invariants to the offending certificate's signature. Matching these two sets of information is computationally expensive, since a complete search must be made by the CA of all certificates of all vehicles, and for each tested signature-view invariant, two shared values need to be multiplied in a privacy preserving way [29]. Thus, to revoke a single vehicle, all vehicles' signature-view invariants must be searched, requiring them to be revealed to a single CA. Consequently, no more privacy from the CA is provided by this scheme as compared to ours. This method also results in larger CRLs as compared to our mechanism, since revoking a vehicle requires adding each if its certificates to the CRL.

Online certificate status protocols (OCSPs) serve as an alternative to distributing CRLs. For an OSCP to be practical for VANET, each vehicle would need a constantly-available connection to the CA. Until DSRC can provide continuous connectivity between every vehicle and the CA, use of an OCSP would require a second wireless technology, e.g., cellular data, to be integrated into the DSRC system.

Papadimitratos et al. [30], propose a mechanism for distributing CRLs from RSUs, where they break the CRL into pieces that are encoded using erasure or fountain codes so that cars that obtain a certain subset of the pieces of the CRL can fully reconstruct the CRL. The authors investigate the bandwidth overhead required by their scheme. However, they do not support their results by vehicle simulation or other empirical data, and the obtained results are based on an assumption of RSU placement density. Specifically, the authors' simulate RSUs placed every 1, 2, and 3 km.

Laberteaux et al. [1], present a mechanism for distributing CRLs through C2C communication. The authors show that this mechanism reduces the number of RSUs required to distribute a CRL (or CRL update if our optimization is used) throughout a metropolitan area. However, the authors do not provide any mechanisms that reduce or address the overhead of their scheme.

7. CONCLUSIONS

We have made two contributions in this paper. First, we proposed a certificate organization method where certificates for a single vehicle are related by a single, secret revocation key. Without this key, certificates are difficult to group, thereby preserving the privacy of a vehicle. However, a revoked vehicle's certificates can be easily identified once the revocation key is distributed via a CRL. To revoke a new vehicle, the CRL need only increase in size by one revocation key, regardless of the number of certificates provided to the revoked vehicle. We presented specific privacy properties of this scheme. We have also shown the real-world performance of our certificate identifier storage mechanism, a Bloom filter. Our analysis has shown that there is relatively low overhead, in terms of memory cost and required computational power, in using our storage mechanism.

Second, we have analyzed and improved the practicality of distributing CRLs. We proposed a mechanism for passing CRL updates, rather than the entire CRL, which reduces the imposed network overhead and is similar to delta CRLs.

Together, these contributions demonstrate that a lightweight privacy-preserving method for VANET security is possible, even in the case of sparse roadside infrastructure.

8. ACKNOWLEDGEMENTS

We would specifically like to thank Jerry T. Chiang for his helpful comments and input in developing some of the ideas and assessments presented in this paper.

References

[1] K. P. Laberteaux, J. J. Haas, and Y.-C. Hu, "Security certificate revocation list distribution for vanet," in *VANET '08: Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking*, (New York, NY, USA), pp. 88–89, ACM, 2008.

[2] P. Golle, D. Greene, and J. Staddon, "Detecting and correcting malicious data in vanets," in *VANET '04: Proceedings of the 1st ACM international workshop on Vehicular Ad hoc networks*, (New York, NY, USA), pp. 29–37, ACM, 2004.

[3] IEEE, *IEEE 1609.2-Standard for Wireless Access in Vehicular Environments (WAVE) - Security Services for Applications and Management Messages, available from ITS Standards Program.*

[4] R. Resendes, "The New 'Grand Challenge' — Deploying Vehicle Communications, Keynote Address — The Fifth ACM International Workshop on VehicluAr InterNETworking (VANET 2008)," September 2008.

[5] M. Raya, P. Papadimitratos, I. Aad, D. Jungels, and J.-P. Hubaux, "Eviction of misbehaving and faulty nodes in vehicular networks," *Selected Areas in Communications, IEEE Journal on*, vol. 25, pp. 1557–1568, Oct. 2007.

[6] D. Cooper, "A More Efficient Use of Delta-CRLs," in *IEEE Symposium on Security and Privacy 2000*, pp. 190–202, May 2000.

[7] L. Blincoe, "The Economic Impact of Motor Vehicle Crashes, 2000," tech. rep., May 2002.

[8] Toyota, "Toyota safety: Toward realizing zero fatalities and accidents." http://www.toyota.co.jp/en/safety_presen/index.html, 2004.

[9] NHTSA, "Traffic safety facts," tech. rep., 2003.

[10] CARE, "Community road accident database." http://europa.eu.int/comm/transport/care/, 2004.

[11] "Minutes of IEEE DSRC Standards Group meetings." http://www.leearmstrong.com/DSRC/DSRCHomeset.htm.

[12] *Standard Specification for Telecommunications and Information Exchange Between Roadside and Vehicle System – 5GHz Band Dedicated Short Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, September 2003.

[13] U. F. C. Commission, "Usfcc report and order fcc 03-324," tech. rep., December 2003.

[14] A. Carter, "The Status of Vehicle-to-Vehicle Communication as a Means of Improving Crash Prevention Performance," tech. rep., 2005.

[15] *Vehicle Infrastructure Integration National System Requirements*, 2008.

[16] V. S. Communications, "Vehicle safety communications project-final report," tech. rep., April 2006.

[17] P. B. et al., "The arbitron national in-car study," tech. rep., December 2003.

[18] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[19] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 281–293, 2000.

[20] "OpenSSL." http://www.openssl.org, September 2008.

[21] B. Schneier, *Applied Cryptography*. John Wiley & Sons, 1996.

[22] Q. Xu, "Vehicle-to-Vehicle Messaging in DSRC," in *First ACM Workshop on Vehicular Ad Hoc Networks(VANET)*, 2004.

[23] G. Korkmaz, E. Ekici, F. Ozguner, and U. Ozguner, "Urban Multi-Hop Broadcast Protocol for Inter-Vehicle Communication Systems," in *In Proc. of the 1st ACM Workshop on Vehicular Ad hoc Networks*, (Philadelphia, USA), October 2004.

[24] B. Parno and A. Perrig, "Challenges in securing vehicular networks," in *Workshop on Hot Topics in Networks (HotNets-IV)*, 2005.

[25] Y. Hu and H. J. Wang, "Location privacy in wireless networks," (Bejing, China), ACM, April 2005.

[26] M. Raya and J. Hubaux, "The security of vehicular ad hoc networks," in *Workshop on Security of ad hoc and Sensor Networks*, 2005.

[27] M. Jakobsson and M. Yung, "Distributed "Magic Ink" Signatures," in *In Eurocrypt '97, LNCS 1233*, pp. 450–464, Springer-Verlag, 1997.

[28] L. Fischer, A. Aijaz, C. Eckert, and D. Vogt, "Secure Revocable Anonymous Authenticated Inter-Vehicle Communication (SRAAC)," in *Proceedings of the 4th Annual Conference on Embedded Security in Cars (escar 2006)*, is-its, November 2006.

[29] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust Threshold DSS Signatures," in *Advances in Cryptology — EUROCRYPT '96*, Springer Berlin / Heidelberg, 1996.

[30] P. P. Papadimitratos, G. Mezzour, and J.-P. Hubaux, "Certificate revocation list distribution in vehicular communication systems," in *VANET '08: Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking*, (New York, NY, USA), pp. 86–87, ACM, 2008.

[31] P. D. MacKenzie and M. K. Reiter, "Two-Party Generation of DSA Signatures," in *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, (London, UK), pp. 137–154, Springer-Verlag, 2001.

Table 4: Parameter definitions for certificate signing process

$\{a\}_b$	Sign/encrypt a using private/public key b (asymmetric cipher)
$E_s(r)$	Encrypt r using key s (symmetric cipher)
$K^+_{CA_j}$	CA j's public key
$x_1 \oplus x_2$	XOR x_1 and x_2
y_r	The r-th certificate identifier
$y_{r,j}$	Section of y_r sent to CA j for revocation search filtering (CA_j's search field)
η_r	Unique revocation search filtering nonce
$z_{r,j}$	Blinded y_r that must be combined by the CAs to revoke a certificate
$\beta_r()$	V's blinding function, parameterized by r

APPENDIX

Appendix A: Multi-CA Certificate Organization

A.1 Privacy-Preserving Certificate Groups

We now propose a method for creating a group of certificate IDs for one vehicle V that appear to be unrelated (privacy preserving), but in fact are related by a single secret value, s, collectively known only to two CAs; that is, no single CA knows s exactly, but two CAs can recreate s by combining the information they each have individually. Our scheme employs two-party blind signatures, as proposed by MacKenzie and Reiter [31]. The revocation key, s, is split between two CAs such that no single CA can compromise a vehicle's privacy. We use two CAs because the cryptographic mechanism we make use of allows only two parties to generate the blinded signatures. For the discussion below, we present a concise overview of the definitions of our parameters in Table 4.

A vehicle V that wants to load itself with certificates begins by first generating B sets of M messages, where M is the number of certificates the car desires, and B is one greater than the number of messages the CAs will require V to unblind as proof that V is not creating malformed messages. For the following discussion, the index r is defined as $r \in [1, M]$ and denotes V's request for its r-th certificate. Each of B sets of messages consists of

$$\left[\{x_{d,1}\}_{K^+_{CA_1}}, \{x_{d,2}\}_{K^+_{CA_2}}\right], \forall d \in [1, B]$$

and M certificate requests. Each individual certificate request is composed as, (in the following, d subscripts denoting each of B requests are dropped for readability.)

$$\left[r, \beta_r\left(y_r, K^+_{V,r}\right), \{rev_{r,1}\}_{K^+_{CA_1}}, \{rev_{r,2}\}_{K^+_{CA_2}}\right]$$

where $y_r = E_s(r), \forall r \in [1, M]$ and $s = x_1 \oplus x_2$. $x_j, j \in \{1, 2\}$ are the keys that V uses to generate the certificate identifiers y_r. Both x_1 and x_2 are required to produce each y_r. Each x_j is encrypted with the respective CA's public key. $\beta_r(y_r, K^+_{V,r})$ is the blinded certificate that V wants the CAs to sign, where $K^+_{V,r}$ is the public key for which this certificate is issued. β_r is the blinding function that V uses to blind the certificate information. $rev_{r,j}$ is the revocation information that CA_j will store concerning V's r-th certificate, where $rev_{r,j} = [z_{r,j}, y_{r,j}, \eta_r]$. $z_{r,1} \oplus z_{r,2} = y_r$, thus $z_{r,j}$ is information the CAs must combine to recover y_r during revocation, which we will describe in more detail below. For each CA, a contiguous range of bits in y_r is selected so that the ranges, of length λ, do not overlap (e.g., CA 1 corresponds to bits 1-8 and CA 2 corresponds to bits 9-16 in each y_r). These ranges are denoted as $y_{r,j}$. Thus, V reveals different parts of its certificate identifiers to each of the CAs in $y_{r,j}$[8]. We will call $y_{r,j}$ CA j's search-field. η_r is a nonce chosen by V, which the CAs will use to associate key data for revocation purposes, as described below. η_r must be unique to the network. If V chooses a non-unique η_r, V regenerates another set of B blinded certificates for signing.

After V presents this information to the CAs, the CAs request that V unblind $B - 1$ sets of requests in order for V to show that it has correctly formed the requests with high probability. The CAs then use a threshold signing mechanism to sign the blinded certificate requests [31] and return the results to V. Each CA stores all of the $\eta_r, y_{r,j}, z_{r,j}$ values so that they can perform revocation.

Our method requires that certificates include a new field, which will hold the r values. The size of this field is the size of one block if a block cipher is used to generate y_r (e.g., 16 bytes for AES).

Privacy Properties.

Considering privacy from a non-CA attacker, the desired privacy property is that it should be difficult for such an attacker to relate the certificates belonging to one vehicle (i.e., one certificate group) without knowledge of s. We now prove that without either key s_1 or s_2, the vehicles owning $(E_{s_1}(r), r)$ and $(E_{s_2}(r), r)$ are indistinguishable. To make this association, the attacker must be able to break the block cipher or PRNG used to generate $E_s(r)$.

LEMMA A.1. *Assume the block cipher is a* pseudo-random permutation, *and two cars, V_1 and V_2, have keys $s_1 \neq s_2$, respectively. We define b to be the block length of the block cipher used or the size of the output from the PRNG, and M to be the total number of certificates assigned to a vehicle by the CA; the attacker will have M or fewer certificates from any vehicle. Suppose that the attacker has*

$$(E_{s_1}(a_1), a_1), \ldots, (E_{s_1}(a_D), a_D)$$

and

$$(E_{s_2}(c_1), c_1), \ldots, (E_{s_2}(c_D), c_D)$$

where $D \leq M$. In the worst case, suppose the attacker can through some means determine that another certificate with $(E_{s_j}(x), x)$ belongs to either V_1 or V_2. Wanting to remove all privacy, the attacker wants to determine to which vehicle $(E_{s_j}(x), x)$ belongs, that is, is $j = 1$ or is $j = 2$. Then, when j is chosen at random to be either 1 or 2, for a randomly chosen x (an attacker wants to determine the origin of a random certificate), an attacker given $(E_{s_j}(x), x)$ gains no advantage on j except with very small probability (Cases 1 and 2 below), or by doing work proportional to the security factor of the block cipher. Specifically, with probability at least $1 - \frac{4D}{2^b}$, an attacker cannot gain any advantage on j except that allowed by the imperfection of the block cipher.

PROOF. We define three cases:

Case 1: If $x \in \{a_q\} \cup \{c_q\}, \forall q \in [1, D]$, then the attacker can trivially determine j with the following algorithm: $x = a_q$ without loss of generality; and $j = 1$ if $E_{s_j}(x) = E_{s_1}(x)$, otherwise $j = 2$. In other words, if x matches one of the certificates it has for V_1 and V_2, then all the attacker needs to do is match $E_{s_j}(x)$ to either $E_{s_1}(x)$ or $E_{s_2}(x)$. Case 1 happens with probability at most $\frac{2D}{2^b}$ and this probability occurs when $a_q \neq c_q, \forall q \in [1, D]$.

Case 2: Case 1 does not hold, and $E_{s_j}(x) \in \{E_{s_1}(a_q)\} \cup \{E_{s_2}(c_q)\}, \forall q \in [1, D]$, then again the attacker can determine j with the following algorithm: $j = 2$ if and only if $E_{s_j}(x) = E_{s_1}(a_q)$ for some q. In other words, by process of elimination, the attacker determines that $j = 2$ because $E_{s_j}(x) = E_{s_1}(a_q)$,

[8]The length, λ, of $y_{r,j}$ should be chosen such that $2^\lambda < M$.

but because Case 1 does not hold, $x \neq a_q$. Essentially, the attacker matches the certificate ID $E_{s_j}(x)$, but the certificate number x does not match a_q and the attacker, as assumed, has already eliminated all other vehicles besides V_1 and V_2, thus, only V_2 remains a possibility. Case 2 happens with probability at most $\frac{2D}{2^b}$ and this probability occurs when $E_{s_1}(a_q) \neq E_{s_2}(c_q), \forall q \in [1, D]$.

Case 3: Neither Case 1 nor Case 2 holds, that is, $x \notin \{a_q\} \cup \{c_q\}, \forall q \in [1, D]$ and $E_{s_j}(x) \notin \{E_{s_1}(a_q)\} \cup \{E_{s_2}(c_q)\}, \forall q \in [1, D]$. We will show that the attacker cannot determine j other than by pure chance, by doing work proportional to the security factor of the block cipher, or by finding a flaw in the block cipher.

Replace the block cipher with random permutations, so we substitute R_1 for E_{s_1} and R_2 for E_{s_2}. Then the attacker must find j given $(x, R_j(x))$, where neither Case 1 nor Case 2 holds. Recalling the notation $A \backslash B = \{a : a \in A, a \notin B\}$, then R_1 is a random bijection of inputs the attacker does not possess ($[0, 2^b - 1] \backslash \{a_q\}$) to outputs the attacker does not possess ($[0, 2^b - 1] \backslash \{R_1(a_q)\}$), and likewise R_2 is a random bijection of inputs the attacker does not possess ($[0, 2^b - 1] \backslash \{c_q\}$) to outputs the attacker does not possess ($[0, 2^b - 1] \backslash \{R_2(c_q)\}$). Since R_1 and R_2 are random functions, and either of them could map an input x from $[0, 2^b - 1] \backslash (\{a_q\} \cup \{c_q\})$ to an output $R_j(x)$ on $([0, 2^b - 1] \backslash \{R_1(a_q) \cup R_2(c_q)\})$, both R_1 and R_2 are equally likely to map x to $R_j(x)$. Thus, an attacker can do no better than pure chance at determining whether $j = 1$ or $j = 2$; that is, the attacker's advantage on j is zero.

Having established that the attacker cannot do better than pure chance when the block cipher is a random permutation, now consider the case where the block cipher is a pseudo-random permutation. We give a proof by contradiction. Suppose there was an attack that compromised privacy when the block cipher is a pseudo-random permutation. Then either that attack would involve doing work proportional to the security factor of the block cipher (in which case Lemma A.1 is vacuously true), or the attack could be used to distinguish the block cipher from a random permutation, contradicting the assumption that it the block cipher is a pseudo-random permutation.

Consequently, the attacker gains no advantage with probability $1 - \frac{2D}{2^b} - \frac{2D}{2^b} = 1 - \frac{4D}{2^b}$, that is, an advantage is gained by the attacker only when Case 1 or Case 2 are true. □

COROLLARY A.2. *After the attacker has observed a set of D certificates from a set of vehicles and then is given a new certificate, the attacker has less than probability $\frac{2D}{2^b}$ (where b is the security factor) to be able to associate this new certificate with or disassociate this new certificate from any previously observed vehicles, unless the attacker performs work proportional to the security factor of the block cipher.*

PROOF. The fact that this certificate is new implies that the attacker has not yet observed this $(E_{s_j}(x), x)$ pair. If the attacker previously heard D certificates, the probability that the x value will match a previously heard certificate from another vehicle, V_{D1}, is less than $\frac{D}{2^b}$ (similar to Case 1 of Lemma A.1). In this case, the attacker could disqualify V_{D1} as the owner of the new certificate. Further, the probability that the $E_{s_j}(x)$ value will match a previously heard certificate from another vehicle V_{D2} is likewise less than $D2^{-b}$ (corresponding to Case 2 of Lemma A.1). In this case, the attacker could disqualify V_{D2} as the owner of the new certificate. Therefore, the probability that the attacker can disqualify any vehicle based on Case 1 and Case 2 of the proof of Lemma A.1 is less than $\frac{2D}{2^b}$. Furthermore, Lemma A.1 shows that in any other case (Case 3), an attacker who does not do work proportional to the security factor of the block cipher cannot do better than pure chance at associating a $(E_{s_j}(x), x)$ pair to a node, or disassociating the pair from a node. □

Corollary A.2 demonstrates that the proposed certificate organization provides near-perfect privacy from other users.

CA Privacy Preservation.

Revealing $y_{r,j}$ to the respective CAs does not reduce a vehicle's privacy. Consider choosing $\lambda = 8$ and $M = 25{,}000$, as discussed above. Now suppose CA_j wants to discover one complete certificate identifier for some unrevoked vehicle. CA_j might do this by listening to traffic on the network and obtaining a certificate that has ID y_r. Next, CA_j wants to eliminate the vehicles this certificate cannot belong to whose certificate setup information CA_j holds. CA_j then reads its $y_{r,j}$ field from the certificate ID. The expected number of vehicles that possess a certificate that has the set of bits in this field identical to the one in the received y_r for its r-th certificate is,

$$E[possess(V, y_{r,j}, r)] = \frac{N}{2^\lambda} = 390{,}625$$

Thus, with high probability, for correct choices of λ and M, the CA will not be able to hear a certificate and meaningfully match it to any certificate creation information that it holds.

Colluding CAs.

If two or more CAs collude, they are able to substantially compromise the privacy of a node. For example, if two CAs collude, then they can match the 2λ bits of each y_r. Continuing the example from above, when $\lambda = 8$, the two CAs could together determine that the car it heard was not one of

$$E[possess(V, y_{r,1}, y_{r,2}, r)] = \frac{N}{2^{2\lambda}} = 1525$$

vehicles (using the same technique as described in the previous paragraph). This is a very small fraction of the population and represents a substantial reduction in privacy.

A.2 Revocation Method

When a certificate with ID y_r is presented to the CAs for revocation, each CA matches its own search field in y_r to each $y_{r,j}$ it has. For each match, CA j adds the corresponding η_r to a list. This search can be efficiently performed by simply keeping an array of pointers to lists, where the array index corresponds to each possible $y_{r,j}$, and the lists pointed to hold all the recorded sets of $[z_{r,j}, y_{r,j}, \eta_r]$. If there are N vehicles in the network (thus N unique η_i's), using a $y_{r,j}$ of length λ reduces the search space from N to $\frac{N}{2^\lambda}$. The CAs then intersect their lists of η_r's. The results of this intersection are the remaining possible η_r's, and the search space has been reduced by $2^{2\lambda}$ (e.g., approximately 1,000,000 times for $\lambda = 10$). Each CA then finds the $z_{r,j}$'s for the remaining η_r's and xor's them together to recover the y_r's. The recovered y_r's are then searched to find the one that matches the y_r in the certificate that is to be revoked. When the match is found, the CAs share their x_j values and compute the revocation key $s = x_1 \oplus x_2$. This key is then added to the CRL and the offending vehicle is revoked.

In combining the $z_{r,j}$ values, vehicle privacy from the CAs is only marginally reduced by a factor of $2^{-2\lambda}$ (e.g., 1 in 65,536 for $\lambda = 8$).

When a vehicle receives a new CRL that includes a new secret s, it proceeds to add revoked certificate IDs to its Bloom filter, as in Section 3.4.