Implementation and Evaluation of Certificate Revocation List Distribution for Vehicular Ad-hoc Networks

Petra Ardelean
advisor: Panos Papadimitratos

January 2009

Abstract

Vehicular Ad-hoc Networks (VANETs) were designed to provide safety and comfort for passengers. Security measurements have to be taken for guaranteeing these. The security consists in making use of asymmetric cryptography for signing messages sent out by the vehicle. But cryptographic material can be compromised or a misbehavior can occur due to some errors. For these reasons the VANET entities credentials should be revoked. The Certificate Authority (CA), for this reason, will make Certificate Revocation Lists (CRLs) and send them to the vehicles. We propose here an implementation of CRL distribution mechanism and the experimental evaluation of the resulted system.

1 Introduction

VANET's main functionalities are safety and entertainment application. Safety is primary relying on information got from neighbor vehicles. Examples of such are speed and location. But for taking them into account the vehicle has to know that they are correct. For this reason asymmetric cryptography is used. Each vehicle has a secret key and publicly known key. The last one's authenticity is certified by a Certificate Authority (CA). Sometimes nodes can be faulty or captured by an attacker, therefor the cryptographic credentials should be removed. One solution is to create CRLs, which contain the identifiers of all certificates that should be revoked. After being created by the CA, it should find a way to send them to the vehicles. Multiple solutions were proposed on how to send the CRLs to the vehicles.

A solution is to relay only in the Road Site Units (RSUs) for the distribution of the CRLs. Each RSU is broadcasting the CRL to the vehicles which are its neighborhood. A variant of this solution is to divided the CRL into redundant pieces and send these pieces to the vehicle via the RSUs [1]. Another proposal is to let vehicles broadcasting the CRLs and rely to the RSUs only for starting the dissemination of the CRLs. This solution is a good approach for systems with weekly deployed infrastructure. In [3] an optimized version is used, where vehicles broadcast only CRL updates to their neighbors. The implementation of the former idea is the scope of this project.

2 Main idea

The focus of this project is to implement the schema proposed by P. Papadimitratos, et al, described in [1] and to do experimental evaluation of the system. The overall schema can be found in Figure 1. A schematic description of

the system is given below.

First, the CA generates the CRL and divides

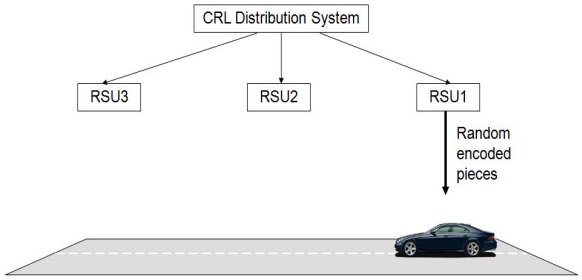

Figure 1: General schema

it into M equal length pieces. These pieces are encoded, using an erasure code, into N redundant pieces. Each piece has a header added, then it is signed by the Certificate Authority (CA). The header contains the CRL version, time stamp (for avoiding reply attacks), the sequence number of the encoded piece and the CA's ID. Figure 2 gives a visual representation of the method. After this step the new pieces are send to the RSUs, which are broadcasting them to the vehicles.

When receiving one of these signed packets the vehicle first verifies the timestamp of the message, then the signature. The signature verification is done by searching in its database the public key associated to the CA ID extracted from the message. If the signature is valid then the vehicle verifies if it has already this piece stored, if not it stores the piece with the associated sequence number. When it has enough pieces it will decode them and obtain the original CRL.

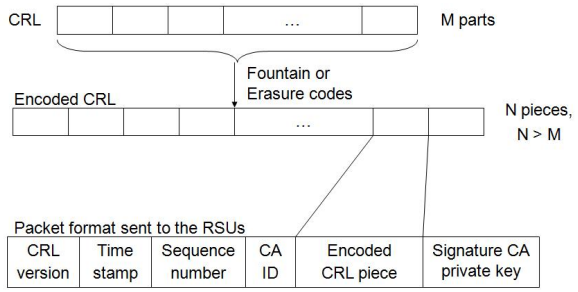

Figure 2: Encoding the CRL

3 Implementation details

The CRL Distribution System implementation is made in the C++ programming language, using the OpenSSL[5] cryptographic library. We Summer that the necessary vectors used for encoding and decoding are preinstalled both in the CA and the vehicles.

A modular approach is used for the implementation of the CRL Distribution System. It is compound of the following two modules: CA and Network Communication (Figure 3). The *CA* is responsible for:

- **Generating certificates and CRLs.** The CA generates X509 certificates for the vehicles. If some vehicles are faulty or their cryptographic material were compromised the CA creates a CRL for the certificates owned by these vehicle. These are X509 CRLs.
- **Encoding the generated CRLs.** The encoding is done by using the Rabin's algorithm as an erasure code[4]. The description of the method is given bellow.
- **Signing encoded piece** using the ECDSA cryptographic algorithm.

The *Network Communication* has the knowledge of the network. It has information about the number of the RSUs, how to get to them and how many pieces it has to send to each of them. It reads a configuration file in which there are written the IP addresses of the RSUs and stores them in a structure. Loose source routing is used for sending different pieces to these IPs. These ensures that each RSU gets only a random subset of the N encoded pieces. When the RSUs get the packet they will broadcast it to the network. The UDP protocol was chosen for encapsulating these messages because of its simplicity.

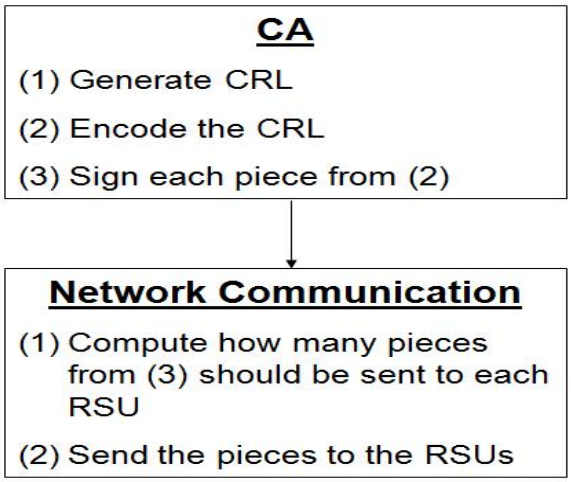

Figure 3: CRL Distribution System

Encoding and Decoding

Robin's algorithm is used for giving redundancy. The original message is viewed as a stream of m bytes. The operations are made in the Galois field mod p. A matrix A_{NxM} is build with N vectors of dimension M, with the property that M of them are linearly independent.

For *encoding* the message is divided into L pieces of length M, with padding if necessarily. These pieces are arranged as a column in a matrix B_{MxL}. Then the $W_{NxL} = A_{NxM} X B_{MxL}$ is computed. Each row in the W matrix is a redundant piece.

The input is the message cut into pieces and the output are N redundant pieces. It is enough of having M pieces (M<N) to decode the original message. The redundancy ratio is N/M.

A more complex description of the algorithm can be found in [4]. The Galois field operations used are implemented by Arash Partow[6] and the matrix inversion is made with the Gauss-Jordan elimination with pivoting method. This source code can be found at [7].

4 Experimental Evaluation

The experiment consists in one machine running the CRL distribution system, one Access Point (AP) playing the role of the RSU and one laptop equipped with wireless card playing the role of a vehicle (see Figure 4). The laptop's configuration is showed in Table 1. The AP rate is 5.5 Mbps.

The goal of the experiments is to determine how much time it takes to a vehicle to get the original CRL (measured from the time the first encoded piece is received) when varying the CRL size and also when changing the encoding vectors (more explicitly varying N and M). From a theoretical point of view this value can be computed as: $t_{getCRL} = M * (t_{vs} + t_{transmit}) + t_{decode}$. Where t_{vs} is the time needed to verify the signature of a received encoded piece, $t_{transmit}$ is the transmission time over the wireless channel and t_{decode} is the time needed to decode the pieces.

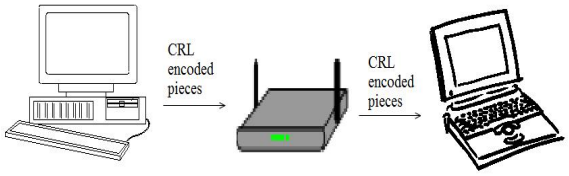

Figure 4: Experimental settings

CPU	Intel 2 GHz
Operating System	Linux
Libraries	openSSL 0.9.8g
C compiler	gcc 4.1.2
Wireless card	802.11g

Table 1: Laptop's configuration

The results for the settings described above for CRL sizes of 200 B, 350 B and 500 B and N=4, M=3 and N=11, M=10 are displayed in Figure 5. It can be observed that with a small value of M the time to get the CRL is smaller because there are just a few pieces which are

transmitted over the medium and need signature verification at the vehicle side. The drawback of using a small M is when having large CRLs, while the length of one piece is large and the transmission time over the wireless channel will increase. If the packet length outruns the MTU then IP fragmentation is needed which adds delays. On the other hand if M has a big value and the CRL is small there is the disadvantage that the sent pieces are too small, so the transmission time increases. Figure 6 shows results for CRL sizes of 8100 B, 40100 B and 80100 B and N=51, M=50 and N=101, M=100. We can observe that if the CRL size is large then the values N and M should be larger too. In conclusion N and M should be chosen in accordance with the length of the CRL. Please note that the X and Y axes of the figures are not the same.

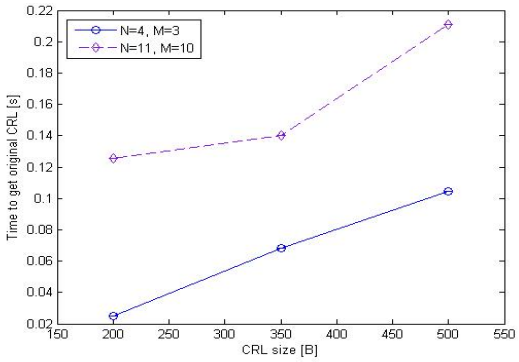

Figure 5: Experimental results for small values of M

5 Conclusions

An implementation of a CRL distribution system for VANET was achieved. With the performance measurements made we can see how the system is performing with different CRL size and different encoding vectors. Even thought the testing system is just a small subset of a real vehicular network it can give us some hints about how this distribution would work in the real system.

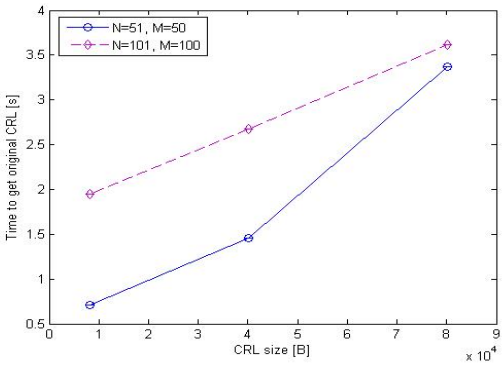

Figure 6: Experimental results for large values of M

6 Further work

One further action consists in making the experiments described in Section 4 but changing the settings by adding two more AP. So, there will be one machine running the CRL distribution system, tree APs playing the role of the RSUs and one laptop equipped with wireless card playing the role of a vehicle. These would be more realistic measurements while they will take into consideration also the movement of the vehicle. Having smaller velocity then a vehicle in the real world is not a problem while the APs' power range is smaller then the RSUs', so the time the laptop spends in the range of an AP is comparable with the time a vehicle would stay in the range of a RSU. After having this results, we want to compare them with the results of the simulations from [1].

An other further action would be to integrate this project into an already existing one. This project implements Secure Vehicular Communications. This project has the implementation of the Certificate Authority (CA), Pseudonym Provider(PP) and vehicles. Vehicles can request certificate and pseudonyms from the CA, respectively PP and also sending to each other secured beacon messages.

Appendix A

How to run the applications

For running the applications it is necessary to have installed an OpenSSL version equal or grater then 0.9.8a and a gcc compiler. Scripts are provided for compiling and running the software. For running the CA use runCA.sh and for the vehicle runVehicle.sh. The contain of these scripts are listed bellow. You can use this as a guide line for compiling and running the applications in the command line.

- The CA
  - the command for compiling
    $g++ -g -O0 -lcrypto$
    $../maths/matrix_inverse/inv.c$
    $../sourceRoute/sourceroute.c$
    $../maths/galoisField/GaloisField.cpp$
    $../vehicle/Vehicle.cpp$
    $../BytesConvertor.cpp$
    $../TimeStamp.cpp$
    $../CryptoModule.cpp$
    $../CodingModule.cpp$
    $CA.cpp../NetworkComm.cpp$
    $Main.cpp -oca$
  - the command for running the vehicle application
    $./ca <networkConfigurationFile>$
    $<codingConfigurationFile>$
    $<IPaddr><socketPortNo>$
    $<privateKeyPath>$
    $<revokedCertificateNo>$
- The vehicle
  - the command for compiling
    $g++ -g -O0 -lcrypto$
    $../maths/matrix_inverse/inv.c$
    $../sourceRoute/sourceroute.c$
    $../maths/galoisField/GaloisField.cpp$
    $../NetworkComm.cpp$
    $../BytesConvertor.cpp$
    $../TimeStamp.cpp$
    $../CryptoModule.cpp$
    $../CodingModule.cpp$
    $Vehicle.cpp$
    $Main.cpp -ovehicle$
  - the command for running the vehicle application
    $./vehicle <socketPortNo>$
    $<CAkeysPath>$
    $<codingConfigurationFile>$

References

[1] P. Papadimitratos, G. Mezzour, and J.-P. Hubaux, *"Certificate Revocation List Distribution in Vehicular Communication Systems"*, short paper, ACM VANET 2008, San Francisco, CA, USA, September 2008

[2] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, and J.-P. Hubaux, *"Secure Vehicular Communications: Design and Architecture"*, IEEE Communications Magazine, November 2008

[3] K. Laberteaux, J. Haas, and Y-C Hu, *"Security Certicate Revocation List Distribution for VANET"*, ACM Mobicom International Workshop on Vehicular Ad Hoc Networks (VANET), San Francisco, USA, September 2008.

[4] P. Papadimitratos and Z.J. Haas, *"Secure Message Transmission in Mobile Ad Hoc Networks"*, Elsevier Ad Hoc Networks Journal, July 2003

[5] *OpenSSL Project*, http://openssl.org/

[6] *Galois Field Arithmetic Library*, http://www.partow.net/projects/galois/

[7] *Gauss-Jordan Elimination*, http://www.dreamincode.net/code/snippet1312.htm