An Embedded Platform for Privacy-Friendly Road Charging Applications

Josep Balasch, Ingrid Verbauwhede and Bart Preneel
K.U.Leuven ESAT/COSIC and IBBT
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium.

Abstract—Systems based on satellite localization are enabling new scenarios for road charging schemes by offering the possibility to charge drivers as a function of their road usage. An in-vehicle installation of a black box with the capabilities of a Location Based Service terminal suffices to deploy such a scheme. In the most straightforward architecture a back-end server collects vehicle's location data in order to extract the correct fees. However, with industry, governments and users being more and more aware of privacy issues the deployment of such system seems to be contradictory. Our contribution is the demonstration of a practical and functional road charging system based on PriPAYD [1]. Our black box is built guaranteeing most of the processing of location data in real-time, thus minimizing overheads required to ensure security and privacy. The performance of our software-based prototype is tested and proves that the deployment of a privacy-friendly solution can be achieved within a minimum cost increment compared to existing road charging schemes.

I. INTRODUCTION

The increasing growth of Location Based Services (LBS) [2] in the last years has led to the appearance of a wide and heterogeneous range of applications. Services based on GSM localization have become quite popular among the general public due to the widespread use of mobile terminals. Knowing the location of services nearby (e.g., shops, doctors,...) or accessing real-time traffic information are just some examples of applications being offered nowadays. In the field of car telematics, a new series of applications and services known as Intelligent Transport Systems (ITS) are currently being developed and studied. Several recently finished European projects [3], [4] have researched the viability of the so-called cooperative systems, i.e., systems that involve Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) communication.

Among these Intelligent Transport Systems, road charging solutions based on satellite localization are of great interest. This approach benefits all actors involved. First, users pay an amount of money proportional to the usage of their vehicle. Second, insurance companies are able to offer fairer and targeted tariffs by charging an amount of money proportional to the statistical risk of accident of roads. Third, governments can achieve mobility reduction in congested roads or cities by charging more money in peak hours. And finally, the environment can benefit from a decrease of the pollution levels due to traffic reduction.

A common requirement for satellite-based road charging systems consists in the installation inside the vehicles of a black box. The minimum functionalities of this device are to determine its own location and to communicate with entities outside the vehicle [5]. The former requirement can be achieved by using a Global Navigation Satellite System (GNSS) like GPS, while for the latter a mobile communications network such as GSM can be used.

The most typical and straightforward architecture for these systems employs the black box as a simple relay, i.e., the vehicle’s location data are forwarded to the service provider (or a third party) who is responsible for computing the road fee. The downside in terms of privacy of this architecture is obvious: an external server is able to collect precise location data of all vehicles in the system, which, even anonymized, can be used to infer personal information about drivers [6]. European countries preparing the deployment of Public Road Pricing are taking this issue into account. As an example, the Dutch government has recently issued a proposal document [7] stating that for privacy reasons only aggregate data will be sent to the collecting entity (unless the user gives express consent for sending detailed location data for the provision of an unrelated service). This decision becomes more important as the Netherlands will be the first Member State in the European Union to deploy a Public Road Pricing system.

An elaborated privacy-friendly solution to the aforementioned problem is given by the PriPAYD (Privacy-Friendly Pay As You Drive) system [1]. The main idea behind PriPAYD is simple: the processing and storage of location data is done locally in the black box, and only the final aggregated premium is sent to the service provider. The scheme ensures that the driver is the only one with access to his location data, while giving the service provider the possibility to audit the logs in case of dispute.

Our contribution is to design and build the first practical and
viable implementation of a road charging application based on PriPAYD. We demonstrate that an embedded solution is possible, and that the basic functionalities of the black box can be achieved by using off-the-shelf hardware modules and free-licensed software tools, requiring just a minimum cost increment with respect to other commercial PAYD schemes. Our pure non-fully optimized software implementation performs all operations with a minimum overhead in terms of execution time.

In the remainder of the paper we only consider the PAYD insurance case. However, we note that all design and implementation steps can be easily mapped to any satellite-based road charging application such as Public Road Pricing.

The rest of this paper is structured as follows. In Sect. II we briefly introduce the functionality and security characteristics of PriPAYD. In Sect. III we delineate a design specification, and we present specific implementation details in Sect. IV. A study of the performance is given in Sect. V. We finally conclude in Sect. VI.

II. AN OVERVIEW OF PRIPAYD

The main purpose of Pay As You Drive systems is to charge drivers proportionally to their road usage. For this, location data of vehicles need to be gathered and processed to calculate a personalized insurance fee. The parameters of this conversion are determined by a certain policy chosen by the insurance company.

Following the same reasoning as [1], we denote as current PAYD model the most typical solution being offered by insurance companies nowadays [8], [9]. These systems are centralized, in the sense that the black box transmits all the location data to a central server responsible for computing the premium. The final bill is later sent to the driver, e.g., by traditional post. The key advantages of such system are mainly two. First, the black box does not need to carry out heavy computations or store large amounts of data, keeping production costs at a minimum level. Second, the system becomes simple and easily scalable as all the ‘intelligence’ is put into a central server. However, the main drawback of this architecture is that drivers have to disclose their fine grained location data, which are not only processed but stored in an external database.

The PriPAYD solution is illustrated in Fig. 1. The main difference with respect to current PAYD model is that all processing and storage of location data is done exclusively in the black box, without transmitting this information to any external entity. Here the black box is responsible for computing the fees corresponding to each trip. The results are aggregated in order to obtain the final premium, which is later sent to the insurance company. Note that in this model the ‘intelligence’ is put on the black box rather than in the external server, which means that digital road maps and a policy need to be locally stored for computations, while giving the insurance company the option to send secure updates.

PriPAYD is designed in such a way that the driver is the only actor that has access to his own location data. For this purpose, the authors of [1] propose the use of a dedicated interface from where location data stored in the black box can be securely retrieved by a portable memory device (e.g., a USB stick). Three conditions need to be met: i) the location data records must be signed by the black box to avoid tampering, ii) an authenticated encryption mechanism must be applied over the location data in order to ensure confidentiality and integrity, and iii) the encryption key used by this mechanism must be known exclusively by the black box and the driver.

The first two points are discussed in Sect. IV, while the third point (key management) is solved as follows. The encryption key $K_s$ is split into two subkeys $KS_1$ and $KS_2$ such that $K_s = f(KS_1, KS_2)$, where the function $f$ ensures that $K_s$ can not be obtained without the knowledge of both subkeys [10]. The subkey $KS_1$ is given to the driver directly from the box when he plugs the USB stick, while $KS_2$ is sent by the insurer together with the invoice (e.g., by post). Note that in case of dispute between the driver and the insurance company, the content of the records (genuinely signed by the black box) can be disclosed for auditing purposes.

In this scheme the black box needs to report at some point (e.g., at the end of the month) the final premium to the insurance company. For any message exchanged between these entities confidentiality and integrity must be ensured, so that no attacker can eavesdrop or modify the content of the information. Moreover, an authentication mechanism must be used such that the insurance company can verify the identity of the box. These problems can be solved by using a public key cryptographic scheme. The message $M$ sent to the insurance company has the following form,

$$M = Enc_{Insurer Key}(D, Sig_{Box Key}(D)),$$  (1)

where the data field $D$ contains the final premium and the subkey $KS_2$, among other fields (see [1] for further details).

III. DESIGN SPECIFICATION

Before diving into design details, we need to address some aspects of the implementation. First, throughout the rest of this work we consider a black box with only basic elements,
i.e., additional modules such as movement sensors that can help detecting GPS spoofing or jamming attacks are not taken into account. These attacks are common for both PriPAYD and current PAYD models, thus they equally affect their performances. Second, we only consider what we denote as 

**normal mode of operation**, i.e., all the operations carried out by the black box, from the reception of GPS strings to the sending of the premium to the insurer. Additional operations such as secure updates are not considered for this demonstration.

The design specification is divided into two steps. We start by enumerating the different types of data created, handled and deleted during the normal mode of operation. Then we present and motivate the high-level modules that conform our PriPAYD embedded platform.

### A. Data Flow

The first part of the design consists in specifying which types of data are created and handled by the black box, as well as how these data are securely encrypted or deleted after its usage.

We denote as **GPS data** the set of all positioning strings sent by the GPS receiver during a trip. These strings follow the NMEA 0183 standard specification [11]. Given that the messages are sent at a default rate of 1 string per second and that the average size of one string is 70 bytes, we can estimate a throughput of 70 bytes/second. Once the journey is finished all these GPS data must be locally encrypted, signed and stored in the black box. Note that the size of this storage is proportional to the duration of the trip, e.g., the location data of a one hour trip requires approximately 70 bytes/second * 3600 seconds = 246 kBytes of storage (without compression). We denote the set of encrypted GPS strings as **encrypted GPS data**, and we note that once these data have been created the original unencrypted GPS data need to be securely deleted from the black box.

The **policy** specifies how the premium is calculated given the location data of a vehicle. We decide to use a similar policy as the one offered by Norwich Union [12] in its discontinued PAYD program, i.e., different prices per Km are selected depending on the type of road and time of the day.

The **mapped data** are formed by the strings obtained after the map-matching operation (see Sect. IV-B), and it is temporarily stored in the black box for the duration of the journey. Each entry of the mapped data contains the necessary information to compute the tariff of the trip, i.e., time of the day, type of road and distance increment with respect to the last entry. The time of the day is directly extracted from the GPS data, while the type of road and the distance increment are computed during the aforementioned map-matching operation. Once the premium corresponding to the trip is calculated, the mapped data are permanently deleted from the black box.

The **premium data** contain the aggregated premium corresponding to the journey, and it is obtained as a result of applying the mapped data to the insurer’s policy.

Fig. 2 shows the data flow in the PriPAYD normal mode of operation. For security reasons the following requirements need to be met: i) the driver can only retrieve encrypted GPS data, ii) GPS data and mapped data must be deleted after the trip, and iii) the policy and the digital road map must be protected against modifications by an attacker.

### B. High-level Modules

The high-level elements in our demonstrator required to execute the normal mode of operation are a processing unit, a GPS receiver, a GSM modem, an external memory module, a battery, and a driver interface.

The **processing unit** contains the software responsible for executing the normal mode of operation of PriPAYD, including a cryptographic library. The processing unit is the main element inside the black box in the sense that it controls all communication with the rest of modules. We choose the NXP LPC2388 [13] as processing unit, a 32-bit low power microcontroller with the widely used ARM7TDMI [14] architecture. This microcontroller implements a RISC architecture, it can run at 72MHz, it offers 512kB of on-chip program memory and 98kB of internal SRAM. In order to develop and test our applications we use as benchmark the Keil MCB2388 evaluation board [15], which comes populated with the LPC2388 microcontroller itself and offers a series of driver interfaces such as a USB port.

The other two typical elements inside the black box are the **GPS Receiver**, which collects the vehicle’s location data, and the **GSM Modem**, which allows to establish communications via the GSM network. In order to minimize the number of interfaces used in our development board, we make use of the Telit GM862-GPS [16] module. This device combines both capabilities (GPS receiver and GSM modem) in a single device, and it communicates through a serial port interface.

Due to the restricted internal memory size of the microcontroller, we require an **external memory** module where large static parameters of the system (policy, digital road maps, and encrypted GPS data) can be stored. In our demonstrator we use an off-the-shelf SD Card connected to the Keil MCB2388 evaluation board. Note that some of the requirements specified
in Sect. III-A are difficult to achieve using an SD card, since it can be easily accessed by an attacker to tamper with the data. Although this choice is suitable for the purposes of our demonstrator, we note that any commercial application should secure the external memory module. A possible solution consists in binding the static files (e.g. policy and road maps) with a genuine signature produced by the insurer, in a way that the black box can randomly check if the contents have been tampered with.

One of the key points of our demonstrator is to manage the power consumption of the black box efficiently. We assume that the black box is powered directly from the vehicle during a journey, and that it uses an internal battery when the vehicle is switched off. This characteristic allows us to differentiate between real-time operations and off-line operations. Note that by maximizing the functionalities executed in real-time we are able to minimize the amount of off-line computations, thus saving internal battery.

Figure 3 illustrates these high-level modules, as well as how they interact with each other in order to exchange information.

IV. IMPLEMENTATION

Our implementation approach consists in developing all the functionalities in software, since this provides an easy way to identify possible bottlenecks when running the code. The implementation details are split into two parts. We start by presenting the requirements and design of the cryptographic library, and later we move into the functional aspects of the normal mode of operation.

A. Cryptographic Library

The first step in the implementation is to develop the cryptographic software library. The main requirement we impose is that the overhead produced by the cryptographic operations should be negligible with respect to the overall performance. Therefore, even though our main programming language is C, we use assembly for frequently used routines in order to improve the execution time.

At a high-level, the cryptographic modules required by PriPAYD are:

1) **Authenticated encryption.** This module is required to encrypt the GPS data that need to be stored in the SD card. We choose to implement the CCM [17] mode of operation for symmetric key block cipher algorithms. The CCM mode ensures confidentiality and authenticity of data by combining the techniques of Counter (CTR) mode and Cipher Block Chaining-Message Authentication Code (CBC-MAC) algorithm.

2) **Public Key Encryption Scheme.** This module is required to encrypt the messages sent by the black box to the insurance company. We decide to implement the RSA-OAEP [18] encryption scheme, which combines the RSA [19] cryptographic primitive and the EME-OAEP [20] encoding technique.

3) **Public Key Signature Scheme.** This module is required by the black box to generate genuine signatures. We implement the RSA-PSS [18] signature scheme, which combines the RSA cryptographic primitive and the PSS [21] encoding technique.

At the core of these high-level modules we require the following cryptographic primitives:

- **Symmetric block cipher.** We select the Advanced Encryption Standard (AES) [22] as our block cipher algorithm, with a key length of 128 bits. Since this primitive is used every time the GPS data need to be encrypted, the code is implemented in assembly to achieve a better performance.
- **RSA algorithm.** The RSA cryptographic primitive is used for the encryption of messages sent to the insurer as well as for generation of signatures. We choose 2048 bits as length of the RSA keys, for both the insurance company and the black box. An existing fast implementation of the ACL [23] library is used to implement the RSA algorithm.
- **Hash function.** The hash function is required by both RSA-OAEP and RSA-PSS schemes. We select RIPEMD-160 [24] as our hash function, using the code provided in the specification.
- **Pseudorandom number generator.** This primitive is required by all three high-level cryptographic modules to generate random strings. Since our evaluation board does not contain any physical hardware-based random number generator, we decide to implement the Salsa20 [25] stream cipher with a random input seed of length 64 bytes. The value of this seed is initialized when installing the box, and it is updated every time the stream cipher is used in order to keep the randomness during the lifetime of the black box.

B. Normal Mode of Operation

The second step of the implementation focuses on the normal mode of operation. We first consider an initialization step, and then we divide the main functionalities in four different operations: map-matching, premium calculation, GPS encryption, and send premium.

**Initialization.** This operation includes all the steps necessary to configure the black box for the end user. The following parameters need to be stored in internal memory: i) the RSA key pair of the black box, ii) the RSA public key of the insurance company, iii) the random seed, iv) the symmetric
key \( K_s \), and v) the codes to start the SIM card. The static parameters in external memory (digital road map, policy) are also stored in this step.

Since most digital roads maps as used by commercial GPS receivers are tied to a license, we select the free map database offered by the collaborative project OpenStreetMap [26]. In order to save memory space and more importantly to speed up the map-matching operation, the road map is previously parsed and compressed such that only contains the indispensable fields per entry: latitude, longitude and type of road. With this approach we are able to store the full road map of Belgium in roughly 30Mb, while the original OpenStreetMap XML file has a size of 600Mb.

Since we do not dispose of real data about the statistical risk of accident of roads, we set arbitrary values to implement a policy that nevertheless suit the purposes of the demonstrator. We divide the roads into 5 different types as specified by OpenStreetMap (from highway to residential) and we define 5 time slots during the day (from 0:00-07:00 to 22:00-00:00). Each possible combination of these fields is assigned a different price per Km.

**Map-matching.** This operation is the only one executed in real-time, i.e., while the vehicle is following a journey. It works as follows. We denote the coordinates of an arbitrary entry of the digital road map as \((\text{lat}_a, \text{lon}_a)\). When a new GPS sentence is available, the microcontroller extracts the current coordinates of the vehicle, namely \((\text{lat}_v, \text{lon}_v)\). The distances (in terms of latitude and longitude) between the vehicle’s location and an entry of the database are then given by \(\Delta \text{lat} = |\text{lat}_v - \text{lat}_a|\) and \(\Delta \text{lon} = |\text{lon}_v - \text{lon}_a|\). In order to calculate the distance \(d\) in meters between these two points we use the Haversine Formula [27],

\[
a = \sin^2\left(\frac{\Delta \text{lat}}{2}\right) + \cos(\text{lat}_v)\cos(\text{lat}_a)\sin^2\left(\frac{\Delta \text{lon}}{2}\right),
\]

\[
c = 2\tan^2\left(\sqrt{a}, \sqrt{1-a}\right),
\]

\[
d = Rc,
\]

where \(R\) corresponds to the Earth’s radius (6378160 meters).

If this distance \(d\) is bigger than a certain threshold we consider that the entry in the database does not match the actual location of the vehicle, and it is just omitted. Otherwise, the entry is accepted as the closest point. When this happens, a new string is added to the mapped data (see Sect. III-A).

Clearly, computing \(d\) for each one of the entries of the digital road map cannot be done in real-time with our microcontroller. In order to speed up the process we modify the order of the entries in the road map database such that they are sorted ascendently by latitude. When searching for the closest node, the microcontroller takes the vehicle’s coordinate \(\text{lat}_v\) and executes a search based on a simple bisection algorithm. Each iteration of this algorithm starts by bisecting an interval of the database in two subintervals (in the first iteration the initial interval contains the whole database). The subinterval where \(\text{lat}_v\) lies is chosen as a new interval, and the algorithm starts again. After a certain number of iterations, the last interval contains the entries susceptible to be the closest node. Then, the distance \(d\) is computed for each entry in order to find the closest database entry, i.e., we perform exhaustive search for each entry in the last interval.

**Premium Calculation.** This operation starts at the end of the journey, right after the map-matching finishes its execution. It is therefore the first off-line operation. For each line of the mapped data we extract the time of the day and the type of road. With these values, the corresponding price per Km is selected from the policy. A simple multiplication between this parameter and the distance increment suffices to obtain each subfee of the journey. All these subfees are then aggregated to obtain the final journey’s premium. After this value is calculated, the mapped data are securely deleted from the SD card.

**GPS Encryption.** This operation is also performed off-line, once the premium of the journey has been calculated. As mentioned in Sect. II, the full GPS data must be first encrypted and then signed by the black box. The cryptographic library performs the authenticated encryption by using the CCM mode, while the cryptographic signature is computed by using the RSA-PSS routine with the secret RSA key of the black box. The concatenation of the encrypted data and the signature is stored in the SD card, while the journey’s GPS data are securely deleted.

**Send premium.** This operation is carried out when the vehicle needs to report the final premium to the insurance company. Note that even though this operation is considered off-line, it just needs to be executed at the end of the month. The main computation corresponds to the encryption of the message sent to the insurer, as specified in (1). We first create the message \(D\) with the final premium and the subkey \(K_2\), among others. After this, we generate an RSA-PSS signature of \(D\) and we append it to the message \(D\). The resulting string \(M\) is finally encrypted using RSA-OAEP with the public key of the insurance company, and then it is sent to the insurance company by SMS.

**V. Performance**

In this section we analyze the performance of the operations described in Sect. IV in terms of execution time. Our first requirement is to achieve a fast execution of the map-matching operation. This implies that a GPS string must be completely processed before the next one arrives, i.e., in less than one second. The other requirement it to minimize the overhead of the off-line computations after the trip (i.e., premium calculation and different data encryptions), and at the end of the month (i.e., send premium). The performance of the main operations and routines is presented in Table I for an arbitrary journey of one hour.

In our real-time operation we can see that the map-matching requires (in the worst case) around 24 million cycles to find the closest entry in the database, which corresponds to 0.328 seconds when the microcontroller’s clock is set to 72 MHz. This time is directly related to the value assigned to the threshold, i.e., the threshold determines the amount of entries
TABLE I: Timings of the implementation for an hour journey

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of cycles</th>
<th>Time (at 72MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL-TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Map-matching (max.)</td>
<td>23 648 336</td>
<td>0.328 s</td>
</tr>
<tr>
<td>OFF-LINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Premium calculation</td>
<td>6 919 096</td>
<td>0.096 s</td>
</tr>
<tr>
<td>Encrypt GPS data</td>
<td>769 809 140</td>
<td>10.691 s</td>
</tr>
<tr>
<td>Encrypt premium</td>
<td>464 069 968</td>
<td>6.445 s</td>
</tr>
</tbody>
</table>

in the last subinterval for which exhaustive search is applied. Note that an off-line processing of the map-matching would require almost 20 extra minutes for a one hour journey.

Although our map-matching algorithm meets perfectly our real-time requirement, it is still resource intensive. The main reason is that neither our digital road map nor our access libraries to the SD card are optimized for this task. We nevertheless provide a simple implementation for the sake of completeness, and we note that a commercial application should use a better (and probably licensed) combination of digital road map and search algorithm optimized for GPS tracking.

Looking at the operations that happen off-line at the end of the trip we see that the computational time of the premium calculation is negligible compared to the GPS encryption, which takes around 10 seconds. Note that both premium calculation and CCM mode for authenticated encryption are clearly dependent on the duration of the journey, e.g., the results would be approximately doubled for a 2-hour journey. This is not exactly the case for the RSA-PSS signature generation. Even though there would be an increment of the execution time, the time required to execute the RSA algorithm is larger than the time required to hash the GPS data.

Finally, the last off-line operation that needs to be done at the end of the month refers to the sending of the premium, and its execution takes around 7 seconds. Similarly to the encryption of the GPS data, the main bottleneck corresponds to the generation of the RSA-PSS signature with the secret key of the box. The reason for the timing difference between the execution of RSA-OAEP and RSA-PSS is because the secret key of the box is considerably larger than the public key of the insurer, and this requires a lot more iterations in the RSA algorithm.

VI. CONCLUSIONS

Public Road Pricing will be deployed in European countries in the following years, with privacy requirements becoming a key issue [7]. Therefore it is essential to prove that privacy-preserving road charging solutions are not just theoretical, but feasible. In this work we have presented a practical demonstrator of a road charging application based on PriPAYD [1]. We have shown that most of the processing of location data can be performed in real-time, and that security and privacy requirements can be fulfilled at the price of adding minimum off-line execution time overheads. Our performance results are achieved by using a pure software implementation, hence we consider them as an upper bound rather than optimal. The black boxes used in the current PAYD solutions already include a processing unit, a GPS receiver and a GSM modem. Our PriPAYD-based black box prototype only requires the addition of an external memory module, thus we conclude that the migration to privacy-friendly solutions could be achieved within a reasonable increment of the production costs.

ACKNOWLEDGMENTS

The authors would like to thank Carmela Troncoso for her continuous support during the development of this project. This work was supported by the Interdisciplinary institute for BroadBand Technology (IBBT) and in part by the Concerted Research Action (GOA) Ambiotics 2005/11 of the Flemish Government and by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

REFERENCES